Université de Montréal

Latent Variable Language Models

par Shawn Tan

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des arts et des sciences
en vue de l'obtention du grade de Maître ès sciences (M.Sc.)
en informatique

August, 2018

Résumé

Dernièrement, il y a eu un renouvellement d'intérêts dans l'application de modèles génératifs en compréhension de la langue. Dans ce mémoire, nous explorons l'ajout de variables latentes dans les modèles de langues traditionnels. Dans le chapitre 1, nous introduisons brièvement les modèles de langues, notamment les modèles n-gram et les modèles de langue neuronaux, couramment utilisés de nos jours. Nous présentons également les auto-encodeurs variationnels ainsi que différents moyens d'améliorer leur performance.

Dans le chapitre 2, nous passons en revue les travaux où des modèles à variables latentes sont appliqués en modélisation de la langue. Nous analysons également l'efficacité de plusieurs de ces méthodes. En particulier, nous analysons les modèles de cite bowman2015generating et cite yang2017improved, et les évaluons entre autres sur *Penn Treebank*.

Dans le chapitre 3, nous présentons un article encore non publié: *Generating Contradictions, Entailments and Neutral Sentences*. Dans ce travail, nous encodons des phrases sources dans une distribution latente. Nous manipulons par la suite cet espace afin de générer des phrases correspondant à certaines implications logiques. Malgré nos efforts infructueux, nous croyons que l'utilisation de variables latentes contrôlables est une direction intéressante à suivre.

Dans le chapitre 4, nous concluons avec un bref survol du mémoire et discutons des travaux futurs possibles.

Mots clés: réseaux de neurones, apprentissage automatique, apprentissage profond, modèles génératifs, compréhension du langage naturel, traitement du langage naturel, modèles de langage

Summary

There has been a renewed interest in generative modeling/unsupervised learning for language for downstream natural language understanding tasks. In this thesis, we explore the augmentation of standard language models with latent variables. In the first chapter, we provide a brief introduction of language models, the classical n-gram treatment and the more common Neural Language Models in use today. We also briefly introduce variational autoencoders and the recent work improving upon them.

In Chapter 2, we review work that explores the space where latent variable models and language models intersect. We then empirically analyse the effectiveness of a couple of these methods. In particular, we re-implement the models from Bowman et al. (2015) and Yang et al. (2017), and benchmark them against the Penn Treebank dataset with some experiments of our own.

In Chapter 3, we discuss an ICML submission: Generating Contradictions, Entailments and Neutral Sentences. In this work, we encode source sentences to a latent distribution space and attempt to manipulate it from there to generate sentences corresponding to the given logical entailment. While our efforts are unsuccessful, we believe that enabling controllable latent variable distributions is an interesting direction to pursue.

In Chapter 4, we conclude with a review of the content covered in the thesis, and a higher-level discussion of what possible avenues of future work could resemble.

Keywords: neural networks, machine learning, deep learning, generative modeling, natural language understanding, natural language processing, language models

Contents

Résumé . . . . . **ii**

Summary . . . . . **iii**

Contents . . . . . **iv**

List of Figures . . . . . **vi**

List of Tables . . . . . **viii**

List of Abbreviations . . . . . **ix**

Acknowledgments . . . . . **x**

1 Introduction **1**
1.1 Language Models 2
1.2 Latent Variable Models and Amortised Inference 3
1.2.1 Improving VAEs 5
1.2.2 Evaluation of VAEs 8
1.2.3 Sequential and Hierarchical VAEs 8
1.2.4 Normalising Flows 10
1.3 Variational Interpretations of Dropout for RNNs 11
1.4 Sentence Representations 12

2 Probabilistic Latent Variable Language Models **14**
2.1 Related Work 15
2.2 Implementation Details 16
2.2.1 Optimisation tricks 16
2.2.2 Comparable Language Model Evaluation 18
2.3 Benchmarking Models on Penn Treebank 19
2.3.1 Experimental details 20
2.3.2 Results 21

2.3.3 Discussion & Future Work 25

3 Generating Entailments, Contradictory and Neutral Sentences . 28
3.1 Prologue 28
3.2 Introduction 28
3.3 Related Work 30
3.4 Method 31
3.4.1 Architecture 32
3.4.2 Compression and Retrieval Functions 35
3.4.3 Model Learning 36
3.5 Experiments 37
3.5.1 Baseline Methods 37
3.5.2 Experiment Settings 38
3.5.3 Quality Evaluation 39
3.5.4 Diversity Evaluation 41
3.5.5 Samples 43
3.6 Discussion 44
3.7 Epilogue 46

4 Conclusion 48

Bibliography 50

List of Figures

1.1 Plate notation depiction of the graphical model. 3
1.2 Left: Not using a transformation flows. Right: Using the deep sigmoidal flows 6
1.3 Left 10 images: Seen examples from the MSCeleb dataset. Each row represents a group of images from one identity. 2nd column from right: The mean of the seen examples. Far right column: Visualising the mean of the inferred identity latent variable. (Figure from Tan et al. (2018)) 9
1.4 9

2.1 A latent variable model with an observed variable with an autoregressive structure. This structure is seen in Gulrajani et al. (2016) and Chen et al. (2016). It is also the graphical model for Bowman et al. (2015), $p(\mathbf{x}|\mathbf{z}) = \prod_1^T p(x_t|x_1, \ldots, x_{t-1}, \mathbf{z})$ 14
2.2 The reconstruction loss and entropy over training iterations. The model is a VAE trained on the MNIST dataset. 17
2.3 An example of a 1-dimensional dilated convolution for language modeling. 18
2.4 Groups of sentence samples given the same sample from the prior. 23
2.5 Interpolating two latent variables samples from the prior. 23
2.6 Interpolating in an irregular distribution may resulting in traversing past low probability coordinates. 24
2.7 Left: 2D latent space with a conditional Gaussian approximate posterior. Right: 2D latent space with IAF approximate posterior. . 25
2.8 Hidden states of the RNN as stochastic latent variables, as discussed in Sohl-Dickstein and Kingma (2015). 27

3.1 The conceptual graphical model behind the formulation of our model. The red arrow represents the inference path from ϕ to $\mathbf{z}$. 31
3.2 The architecture of the model. The autoencoder maps given premise $\boldsymbol{\phi}$ to a sentence representation $\mathbf{z}$, and reconstructs $\boldsymbol{\phi}$ from $\mathbf{z}$. Samples are drawn from the prior conditioned on $\boldsymbol{\eta}$ and ℓ. The classifier takes $\mathbf{z}$ and $\boldsymbol{\eta}$ as input, and outputs probability of l. The discriminator takes $\mathbf{z}$, $\boldsymbol{\eta}$ and l as input, and predicts whether $\mathbf{z}$ is given by the autoencoder or the prior. 32

3.3 *Memory Operation Selection Module* takes a pair of vector $(\mathbf{k}, \mathbf{v})$ as input, output a vector $\mathbf{o}$. $\mathbf{k}$ provide the control signal for the layer to compute a weighted sum of candidate weight matrices. The obtained matrix is used as the weight matrix in a normal feedforward layer, that takes $\mathbf{v}$ as input and outputs $\mathbf{o}$. 35
3.4 Different classification precisions given by our classifier in our model (MOSM, N=10) during training. Sample Precision shows the probability that classifier predicts correct label for generated premise and related real hypothesis. Valid precision shows the probability that classifier predicts correct label for real premise and real hypothesis. Z precision shows the probability that the feedforward network $f_{\text{classifier}}(\mathbf{z}, z_{\boldsymbol{\eta}})$ predicts correct label ℓ, for given $\boldsymbol{\eta}$, ℓ and $\mathbf{z}$ drawn from prior $p(\mathbf{z}|\boldsymbol{\eta}, \ell)$. 41
3.5 Visualization of the effect of auxiliary loss with multiple samples. For a pair of $(\boldsymbol{\phi}, \boldsymbol{\eta})$, we repeat 100 times the process of compute auxiliary loss (N=10) in Equation 3.26. Blue points represent $\mathbf{z}_i$ selected by Equation 3.26, green points represent $\mathbf{z}_i$ that are not selected. Our model (MOSM, N=10) is used for computing $\mathbf{z}$ and perplexities. t-SNE is used to visualize high-dimensional data (Maaten and Hinton, 2008). 43
3.6 Example sentence generated by our model (MOSM, N=10). **H** is the hypothesis, **L** is the label, **S1** is the first sample, and **S2** is the second sample. The samples shown below the line are drawn from a model trained without the auxiliary loss. 44
3.7 Traversing the latent space to a common representation 46

4.1 Vauquois Triangle 49

List of Tables

2.1 Results for language model experiments. For both sentence-level and full-corpus evaluation, we breakdown the *test* perplexity and the ELBO breakdown on the *train* set. 22

3.1 Classification accuracies for different state-of-the-art models on our samples. The row labeled RANDOM we randomly permuted the premises of the original test set and ran them through the classifiers to test for the models' reliance on just the hypothesis for classification. 39
3.2 The confusion matrix for the samples from the best model MOSM ($N = 1$) 40
3.3 BLEU score for different models 42

List of Abbreviations

VAE	Variational Auto-Encoder
RNN	Recurrent Neural Network
OOV	Out-of-Vocabulary
LM	Language model
PLVLM	Probabilistic Latent Variable Models
ELBO	Evidence LOwer Bound
MADE	Masked Autoregressive Density Estimator
IAF	Inverse Autoregressive Flow
NAF	Neural Autoregressive Flow
LSTM	Long Short-Term Memory
ILDJ	Inverse Log Determinant of the Jacobian
GAN	Generative Adversarial Networks vMF von Mises-Fisher
(A)SGD	(Averaged) Stochastic Gradient Descent

Acknowledgments

Thank you to Aaron Courville for being a great advisor. Your approach to our discussions have proved useful in deciding directions of research to pursue, and I am grateful for the continued faith in my capabilities. And Chris Pal, who along with Aaron interviewed and accepted me into the programme, and provided encouragement/enthusiasm along the way for the projects I was working on.

I'm thankful,

For the environment that MILA has provided: an excellent group of individuals with the same broad interests in deep learning, but yet diverse enough to find like-minded collaborators in more domain-specific tasks. To Yikang and Chin-wei and Jae Hyun for being great collaborators and sounding boards for new ideas, and greater friends (Sorry for the bouts of gloom and doom you had to put up with). To Joseph for keeping the idealism alive, and pushing me along (and for easing the burden of me being the oldest in the room all the time). To Francis and Philippe, for going through the Maitre es Sciences en Informatique with me (and for teaching me useful Quebecisms). To Alex, for giving me trivia breaks while working at Second Cup, and for giving me his thesis as a reference.

To the people of 3248: Amina, Zhouhan, Mirco, Philemon, Samuel, Alex, Devon, Tristan, Margaux, Francois; thanks for creating a great place to work, discuss, and for tolerating me and my stupid jokes.

To the MILA staff, for the thankless job they've had to do over the years.

And my family, who've had to put up with my various pursuits over the years.

1 Introduction

Natural Language Processing and Understanding is still an active area of research. In spite of the huge growth of labelled datasets for various tasks in recent years, the internet holds a wealth of unlabelled natural language data. More recently, there has been a resurgence of using unsupervised methods as a pre-training step before doing discriminative training on the task of interest.

Radford et al. (2018) pre-train a language model using the Transformer (Vaswani et al., 2017) architecture. The model is then fine-tuned on different supervised learning tasks, and achieves good results on most of the tasks they were tested on.

Peters et al. (2018) takes a different approach, by learning a model that provides a word representation that is contextualised by the sentence it is used in. These embeddings, which they call *Embeddings from Language Models* (ELMo) are then used on supervised learning tasks, where they also perform exceedingly well.

Trinh and Le (2018) train an ensemble of large RNN Language models on the 1 Billion word dataset (Chelba et al., 2013). Through framing the downstream co-reference resolution task as a language generation problem, the authors were able to demonstrate that such language models also encompass some simple reasoning capability about the real-world.

These results suggest the role of the language model does not stop at predicting how likely a given sentence is. The process of training a model of language data seems to, as a side effect, learn other structures in the model that are essential for making a prediction on these downstream supervised tasks.

In this report, we explore one possible method to augment language models to better model higher level contexts: Latent variable models. In the following section we discuss the fundamentals of language modelling and recent related work. We then discuss Variational Autoencoders in Section 1.2, exploring the basics, and summarise recent work in improving VAEs.

1.1 Language Models

Language Modelling is at the heart of many Natural Language Processing related tasks. The goal is to estimate the probability of a sequence of words $p(x_1, \ldots, x_T)$. In tasks like machine translation (Koehn et al., 2003) or speech recognition (Bahl et al., 1990), a language model can aid in selecting the most likely sentence given several possibilities. They are usually factorised sequentially,

$$p(x_1, \ldots, x_T) = \prod_{t=1}^{T} p(x_t | x_1, \ldots, x_{t-1}) \tag{1.1}$$

Additionally, language models are measured with *perplexity*.

$$2^{\frac{1}{N} \sum_{t=1}^{T} -\log_2 p(x_t | x_1, \ldots, x_{t-1})}$$

This is a measure of how well a probabilistic model is at predicting the data, and is essentially an exponentiated cross-entropy. An intuitive interpretation of it is the average number of 'choices' the probabilistic model predicts at every word.

Traditionally, an n-gram model is used. This uses counts of windows of n words, making the assumption that each word is only dependent on its preceding $n-1$ words:

$$p(x_1, \ldots, x_T) = \prod_{t=1}^{T} p(x_t | x_{t-n-1}, \ldots, x_{t-1}) \tag{1.2}$$

The way $p(x_t | x_{t-n-1}, \ldots, x_{t-1})$ is approximated using the occurence counts of the n-grams,

$$p(x_t | x_{t-n-1}, \ldots, x_{t-1}) = \frac{\text{count}(x_{t-n-1}, \ldots, x_{t-1}, x_t)}{\text{count}(x_{t-n-1}, \ldots, x_{t-1})} \tag{1.3}$$

However, consider that with this approach, the size of the count table grows exponentially $O(|V|^n)$, where V is the set of vocabulary of the dataset. Moreover, the sparsity of this table increases as n increases, so many evaluated n-grams do not occur in the training set. Smoothing and back-off (Kneser and Ney, 1995) techniques are used to get around this problem.

A neural language model (NLM) (Bengio et al., 2003) improves upon this by

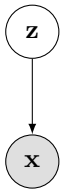


Figure 1.1 – Plate notation depiction of the graphical model.

using a vector-space representation for words and a probability function. This allows words that appear in similar contexts to have relatively similar *embedding*, which is a real vector. Where in an n-gram model the sequence of words would never be observed before, the model can use the learned semantic features of the word in the embedding form to make a prediction. Later, neural language models using recurrent neural networks, or recurrent neural network language models (RNN LM), were used to capture long-term dependencies in sequences (Mikolov et al., 2010).

Since then, different methods of improving language models have been introduced, and steadily improving the state-of-the-art on language modelling. These techniques include improving performance on out-of-vocabulary (OOV) words (Bahdanau et al., 2017), updating parameters given the preceding context (Mikolov, 2012; Jelinek et al., 1991), biasing prediction to recent words (Grave et al., 2016), ensembling the softmax output (Yang et al., 2018), etc.

1.2 Latent Variable Models and Amortised Inference

Latent variable probabilistic models assume the existence of an unobserved variable in the generation of the observed data. This latent variable can be thought of as underlying factors that result in the observed variable. In latent variable models we want the learned model to capture some important characteristics of the observed data. To illustrate with a simple example in Figure 1.1, we have datapoints $\mathbf{x}$, and each $\mathbf{x}$ has an associated, unobserved, variable $\mathbf{z}$ that it is associated with.

If we want to maximise the likelihood of the data over such a model, we have to

marginalise the latent variable,

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})\, p(\mathbf{z})\, \mathrm{d}\mathbf{z} \tag{1.4}$$

However, this is intractable. Variational techniques (Jordan et al., 1999; Wainwright et al., 2008) help with this issue by specifying the *Evidence Lower BOund* (ELBO) that is tractable to optimise over. Variational Autoencoders (Kingma and Welling, 2013; Rezende et al., 2014) then introduced the reparameterisation trick and brought the technique to learning deep generative models. The resulting loss used in Variational Autoencoders is:

$$\log p(\mathbf{x}) = \log \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[p_\theta(\mathbf{x}|\mathbf{z}) \frac{p_\theta(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right] \tag{1.5}$$

$$\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}|\mathbf{z}) \frac{p_\theta(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right] \tag{1.6}$$

$$= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}|\mathbf{z}) \right] - D_{\mathrm{KL}} \left(q_\phi(\mathbf{z}|\mathbf{x}) \,\|\, p_\theta(\mathbf{z}) \right) \tag{1.7}$$

Relating this back to natural language, $\mathbf{x}$ could be a sequence of text (a sentence, perhaps), while $\mathbf{z}$ represents the global context of the given sentence. More concretely, to generate a sentence, we first sample for $\mathbf{z}$, then auto-regressively generate the sentence $\mathbf{x}$. Note that in this way, the variance in samples from the model comes not only from the language modelling part, but also the latent variable.

However, training such a model is fraught with optimisation issues. Consider the information theoretic view of the ELBO:

$$-\mathcal{L}(\theta, \phi; x) = \underbrace{-\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}|\mathbf{z}) \right]}_{\text{Reconstruction loss}} + \underbrace{D_{\mathrm{KL}} \left(q_\phi(\mathbf{z}|\mathbf{x}) \,\|\, p_\theta(\mathbf{z}) \right)}_{\text{Information gain}} \tag{1.8}$$

In some applications, like language models, $p_\theta(\mathbf{z}|\mathbf{x})$ is modelled auto-regressively. In most cases where this can be achieved, there are usualy superficial correlations in each dimension of the output that can be leveraged to accurately predict the original data point (e.g. Neighbouring pixels on an image, structure of language, a frame of speech audio is unlikely to change much in consecutive frames.) Due to the information bottleneck, the likelihood can learn to be conditionally independent of

the latent variable. This, however, does not mean that the data point is best modelled without the latent variable. Chen et al. (2016) observe that only when information cannot be encoded locally by the autoregressive model will the information be encoded in $\mathbf{z}$.

1.2.1 Improving VAEs

In the single-level, factorised Gaussian latent variable initially proposed in Kingma and Welling (2013) and Rezende et al. (2014), there are several aspects that can be modified to improve the VAE.

— **The approximate posterior**

$$q(\mathbf{z}|\mathbf{x})$$

Otherwise called the *encoder*, or the *inference network* in the context of VAEs, where they are parameterised by a deep network (written here as f). The conditional distribution over $\mathbf{z}$ given the data $\mathbf{x}$ is usually defined as Gaussian, with parameters that are a function of $\mathbf{x}$,

$$q(\mathbf{z}|\mathbf{x}) = \mathcal{N}\left(f_\mu(\mathbf{x}), f_\sigma(\mathbf{x})\right).$$

— **The likelihood**

$$p(\mathbf{x}|\mathbf{z})$$

Otherwise called the *decoder*, or the *generative* network in the context of VAEs, where they are also parameterised by a deep network.

— **The prior**

$$p(\mathbf{z})$$

In Kingma and Welling (2013), this is defined $\mathcal{N}(\mathbf{0}, \mathbf{1})$ and not parameterised.

All three aspects of the model can be improved, not only at the level of architecture, but in the types of distributions that are assumed for each of them in the model.

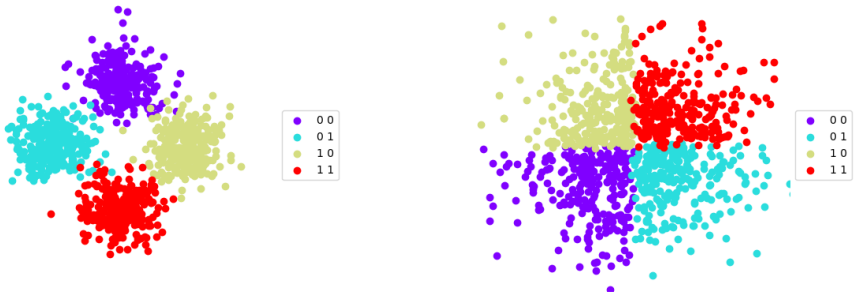


Figure 1.2 – Left: Not using a transformation flows. Right: Using the deep sigmoidal flows

Improving the Likelihood

Consider modelling an image using a single latent variable model. In order to produce a reasonable, realistic looking image, we need to decide the objects in the image, their location, etc. Beyond these abstract ideas, there are also the details, like lighting and texture that we need to model.

When the probability of the given datapoint is defined as being factorised over every dimension, the implication is that the distribution over the observation is assumed to be completely dependent upon the latent variable. However, the latent variable has limited capacity, and may model only the aspects that contribute to most of the reconstruction loss (abstract concepts) and there may be variations at a lower level of abstraction that may depend on other parts of the input (details).

Some examples of such models are Gulrajani et al. (2016) for images, with a slightly more general discussion found in Chen et al. (2016), and Bowman et al. (2015) for text.

Improving the Posterior

Recall again that one of the goals of the generative model is to allow us to sample from the distribution over $\mathbf{z}$, and generate reasonable examples of our data. Ideally, this also means that there has to be a mapping from the data $\mathbf{x}$ to the distribution over $\mathbf{z}$ such that marginalising out the data, we recover the defined prior over $\mathbf{z}$.

Unfortunately, defining the approximate posterior as a Gaussian may not satisfy this requirement well enough. Consider a two-bit example: we sample (x_a, x_b) from

two independent Bernoulli distributions both with probability being 0.5. There are four possibilities. And we want to learn a VAE on this dataset. If we assume the approximate posteriors to be Gaussians (see left of Figure 1.2), then we would end up with an aggregate posterior $q(z) = \sum_{i=1}^{4} q(z_i|x_i)\tilde{p}(x_i)$ being a mixture of Gaussians that fails to be prior-like ($p(z) = \mathcal{N}(z; 0, I)$ in this case). This is because the posterior distributions fail to capture the true ones, and if our inference network $q(z|x)$ is good enough to perfectly model the true posteriors, the aggregate posteriors should be the shape of the prior (see right of Figure 1.2).

There are several methods to approach this problem. Maaløe et al. (2016) proposed using an auxiliary variable conditional on the data and that the posterior is conditioned on. Marginalising the auxiliary variable then allows for a much more flexible posterior distribution. Another popular class of methods are normalising flows, which we will discuss in Section 1.2.4.

Improving the Prior

Just as we can augment the approximate posterior, we can also modify the prior such that it is learned from the data.

The expressivity of neural networks and developments in improving the posteriors allow the data to be mapped really well to simple priors like a Gaussian. So if we can already modify the posterior, why change the prior? One reason may be due to discontinuity when interpolating in the latent space. If we imagine each colour code (See Figure 1.2) to be a different category in the data, moving from one colour group to another will represent a drastic change in the type of data decoded. This means that sampling from close regions in the latent space may not represent close relationships in the data space. Allowing the prior to model a lower density between these different categories may be better when the type of representation learned is more important than simply sampling data points from the model.

Tomczak and Welling (2017) proposes parameterising the prior as a mixture over a sample of the dataset, or using learned *pseudo-inputs*. Serban et al. (2016) introduces a prior parameterised by a piece-wise linear function. In Huang et al. (2017), the authors parameterise the prior with Real NVP transformations (Dinh et al., 2016).

Other ways of achieving this is using MADE (Germain et al., 2015) to learn an autoregressive density model of the prior. MADE is an autoencoder masked in such

a way that each output dimension is conditioned only on the previous dimensions. With this constraint, the output can be interpreted as conditional probabilities of each dimension. The joint probability is then the probability of the data. This property allows us to use MADE as a density model.

1.2.2 Evaluation of VAEs

The evaluation of VAEs would be by estimating the log-likelihood. One method for doing this is using importance sampling as discussed in Rezende et al. (2014):

$$\int p(\mathbf{x}|\mathbf{z})\, p(\mathbf{z})\, \mathrm{d}\mathbf{z} = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[\frac{p(\mathbf{x}|\mathbf{z})\, p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x})}\right] \tag{1.9}$$

$$\approx \frac{1}{K}\sum_{k=1}^{K} \frac{p\left(\mathbf{x}|\mathbf{z}^{(k)}\right) p\left(\mathbf{z}^{(k)}\right)}{q\left(\mathbf{z}^{(k)}|\mathbf{x}\right)} \qquad \mathbf{z}^{(k)} \sim q(\mathbf{z}|\mathbf{x}) \tag{1.10}$$

The method uses K samples from the approximate posterior, and $\frac{p\left(\mathbf{z}^{(k)}\right)}{q\left(\mathbf{z}^{(k)}|\mathbf{x}\right)}$ as a weight for estimation of $p(\mathbf{x})$. Kingma and Welling (2013) propose an alternative method for estimating the likelihood.

Since $\frac{p\left(\mathbf{x}|\mathbf{z}^{(k)}\right)p\left(\mathbf{z}^{(k)}\right)}{q\left(\mathbf{z}^{(k)}|\mathbf{x}\right)}$ will be really small, computing them practically would require performing the computation in log-space.

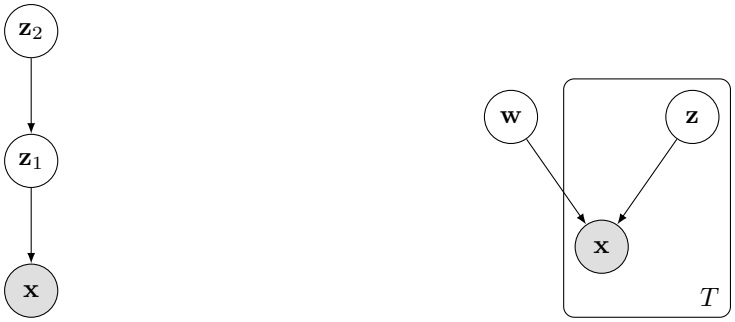
$$\log \frac{1}{K}\sum_{k=1}^{K} \frac{p\left(\mathbf{x}|\mathbf{z}^{(k)}\right) p\left(\mathbf{z}^{(k)}\right)}{q\left(\mathbf{z}^{(k)}|\mathbf{x}\right)} = -\log K + \underbrace{\log \sum_{k=1}^{K} \exp\left(\log \frac{p\left(\mathbf{x}|\mathbf{z}^{(k)}\right) p\left(\mathbf{z}^{(k)}\right)}{q\left(\mathbf{z}^{(k)}|\mathbf{x}\right)}\right)}_{\text{log sum exp}} \tag{1.11}$$

It is interesting to note that the log-sum-exp operation is also sometimes known as the "soft max" ($\log \sum \exp \approx \max$). The evaluation metric can then be viewed as taking many samples (K) from q, and evaluating the single sample Variational Lower Bound, and evaluating the "soft max" of that. Consequently, we should expect high estimates given more samples (larger K) (Burda et al., 2015).

1.2.3 Sequential and Hierarchical VAEs

The single-latent-variable model can be extended to multiple latent variables. Since the VAE framework is a way of training probablistic graphical models with

Figure 1.3 – Left 10 images: Seen examples from the MSCeleb dataset. Each row represents a group of images from one identity. 2nd column from right: The mean of the seen examples. Far right column: Visualising the mean of the inferred identity latent variable. (Figure from Tan et al. (2018))



(a) Hierarchical latent variables **(b)** Shared common latent variable.

Figure 1.4

deep neural networks, we simply have to decide what the structure of the graphical model is.

Chung et al. (2015) first used the VAE framework as a method of training a sequential latent variable model. The approximate posterior was modelled using only previous items in the sequence, but depending on the assumptions made about what the latent variable was supposed to model, this is not a strict requirement. If we consider a single latent variable in the sequence it is not conditionally independent of the future observations. Hence, the true posterior $p(\mathbf{z}_t|\mathbf{x})$ should also be conditioned on the entire sequence. Shabanian et al. (2017) approaches this by modelling the inference network as a Bi-LSTM, encoding information from both 'past' and 'future'.

While a hierarchy of latent variables was proposed in Rezende et al. (2014), no experiments were performed to show its feasibility. Sønderby et al. (2016) experiment with the inference model required to train such a hierarchical latent representation. Gulrajani et al. (2016) combine this with the auto-regressive PixelCNN (Oord et al., 2016), so that the latent variables are also auto-regressive.

The hierarchy can also extend beyond the topology of an image, or a single data point. Edwards and Storkey (2016) and Bouchacourt et al. (2017) presents several ways data points with commonalities can be grouped together for statistical efficiency, sharing one global latent variable which encodes their similarities, while still having individual, per-data point latent variables. Hsu et al. (2017) use a similar approach. They focus on learning speaker level features from different utterances from the same speaker.

Using the information theoretic perspective, we can again look at the objectives of a simplified version of this type of model used in Tan et al. (2018):

$$
\begin{aligned}
&-\log p\left(\mathbf{x}_1, \ldots, \mathbf{x}_T\right) \leq \\
&D_{\mathrm{KL}}\left(q\left(\mathbf{w}|\mathbf{x}_1, \ldots, \mathbf{x}_T\right) \| p\left(\mathbf{w}\right)\right) \\
&\quad + \mathbb{E}_{q(\mathbf{w}|\mathbf{x}_1,\ldots,\mathbf{x}_T)}\left[\sum_i^N D_{\mathrm{KL}}\left(q\left(\mathbf{z}_1, \ldots, \mathbf{z}_T|\mathbf{x}_1, \ldots, \mathbf{x}_T, \mathbf{w}\right) \| p\left(\mathbf{z}_1, \ldots, \mathbf{z}_T|\mathbf{w}\right)\right)\right. \\
&\qquad \left. -\mathbb{E}_{q(\mathbf{z}_1,\ldots,\mathbf{z}_T|\mathbf{x}_1,\ldots,\mathbf{x}_T,\mathbf{w})}\left[\sum_i^N \log p\left(\mathbf{x}_1, \ldots, \mathbf{x}_T|\mathbf{z}_1, \ldots, \mathbf{z}_T, \mathbf{w}\right)\right]\right]
\end{aligned}
$$

If we view the KL-divergence term as a penalty on the information transmitted from the inference model to the generative model, the loss encourages the model to encode as much information as possible in $\mathbf{w}$ as opposed to in $\mathbf{z}_t$, which is penalised as many times as there are images.

1.2.4 Normalising Flows

As mentioned in Section 1.2.1, the vanilla VAE Gaussian approximate posterior may be too naive an assumption in some circumstances. One approach to get around this are *finite normalising flows* proposed in Rezende and Mohamed (2015).

At the heart of the method is an invertible function f, and the Change of

Variables rule. Given an invertible function $\mathbf{z} = f(\mathbf{x})$ with a Jacobian with a tractable determinant: $\left|\frac{\partial f^{-1}(\mathbf{z})}{\partial \mathbf{z}}\right|$, we can transform the original, *simpler*, distribution, a Gaussian for example, to something more complex. Given the probability density of the original distribution $q(\mathbf{z})$ and the determinant of the Jacobian of f we can compute the probability of the transformed $\mathbf{z}$, $\mathbf{z}'$:

$$q(\mathbf{z}') = q(\mathbf{z}) \cdot \left|\frac{\partial f^{-1}(\mathbf{z}')}{\partial \mathbf{z}'}\right| = q(\mathbf{z}) \cdot \left|\frac{\partial f(\mathbf{z})}{\partial \mathbf{z}}\right|^{-1} \tag{1.12}$$

Most of the work surrounding normalising flows deal with searching for invertible functions with a tractable inverse-log-determinant-jacobian (ILDJ). Restricting the functions to be auto-regressive in nature means that the Jacobian is lower triangular and the determinant of a lower triangular matrix is simply the product of its diagonal entries. Kingma et al. (2016) used this technique to create a non-linear invertible flow the authors call *Inverse Autoregressive Flows* (IAF). By chaining several such transforms, and permuting the dimensons (also an invertible transformation with ILDJ = 0), one can form fairly complex transforms. Huang et al. (2018) proposes *Neural Autoregressive Flows* (NAF) which are made up of monotonic functions with autoregressive conditioning, like IAF. Additionally, they demonstrate that the NAF are universal approximators for continuous probability distributions.

Another example of a tractable ILDJ are volume preserving transforms which have a ILDJ = 0. Tomczak and Welling (2016) uses Householder transforms to achieve this, which is a special case of an orthogonal linear transformation.

1.3 Variational Interpretations of Dropout for RNNs

Dropout (Srivastava et al., 2014) is a commonly used regularisation method that has been used very effectively for regularising large networks. Merity et al. (2017) did an empirical study, and demonstrated that using a standard LSTM with a variety of regularisation methods gives good baseline results for language modeling. Among these methods are Variational Dropout (Gal and Ghahramani, 2016) and DropConnect (Wan et al., 2013).

Gal and Ghahramani (2016) provide a theoretical justification for using dropout in RNNs by viewing the parameters (ω) of the RNN function $\mathbf{h}_t = f(\mathbf{x}_t, \mathbf{h}_{t-1})$ as a random variable. Consequently, only one sample of ω is used throughout the sequence. The upshot of this is that during training, the same dropout mask is used throughout the sequence that is being predicted.

Wan et al. (2013) proposes dropping out the weights during training, as a generalisation of dropping out hidden activations. Dropout can be viewed as a special case of DropConnect when dropping out entire rows of the weight matrix. In Merity et al. (2017), DropConnect is applied at the transition functions only, and, like Variational Dropout, the same mask is applied throughout the sequence. Due to implementation constraints, the same mask is also applied for the entire minibatch.

1.4 Sentence Representations

While word representations have given us dramatic performance improvements for language modelling, sentence representations still seems to be an open area of research. There have been some attempts at encoding a sentence into a fixed-length vector, e.g. Skip-thought vector (Kiros et al., 2015). Other researchers fundamentally disagree with the technique [1]. More recent approaches have modelled these sentence representations as sequence of vectors, and used an attention mechanism over these to conditionally generate their output (Bahdanau et al., 2014).

If we consider a vector that encodes the entire sentence perfectly for decoding, then the encoding and decoding process is deterministic. However, a useful vector may require just the encoding of semantics, which may manifest as a sentence using different words, and / or a different sentence structure. This means that for each latent semantic representation, there are various possible sentences associated with each latent code.

1. 'You can't cram the meaning of a whole %&!$ing sentence into a single $&!*ing vector!" – Ray Mooney

In particular, we are interested in distributions over the latent representations of natural language. More formally,

$$p(x_1, \ldots, x_T) = \int p(x_1, \ldots, x_T|\mathbf{z})\, p(\mathbf{z})\, \mathrm{d}\mathbf{z} \tag{1.13}$$

Another property that would be useful is for sentences that are semantically close to be close in the latent space as well. For these distributions to be useful, they should impose some form of continuity in the generated sentences if we take an interpolated point between the latent representation of two sentences. At the writing of this report, this is still a fairly difficult task to achieve.

In the next chapter, we look at one method of learning such representations, and analyse the method quantitatively and qualitatively to see if they fulfil our requirements.

2 Probabilistic Latent Variable Language Models

At the time of this writing, the de facto standard for language modelling is the RNN-LM, which breaks down the language modelling task into a next-word prediction problem. However in doing so, the RNN-LM does not make use of global features that may exist.

This separates the generation of a sentence into a global latent variable $\mathbf{z}$, and the sequence of words $x_1, \ldots, x_T$, conditional on $\mathbf{z}$. Ideally, factorising the probabilistic model this way explicitly models more abstract features of a sentence. The topic (Blei, 2012) of the sentence is a good example of such a higher-level feature that would be useful in sentence generation. In this chapter, we discuss a class of language models we will refer to as Probabilistic Latent Variable Language Models (PLVLM).

We are partial to this method due to its probabilistic interpretation and ease of quantitative evaluation.

Figure 2.1 illustrates the graphical model that one example of such a model would take (Bowman et al., 2015). The latent variable $\mathbf{z}$ represents the information that is shared throughout for all the words through the sentence.

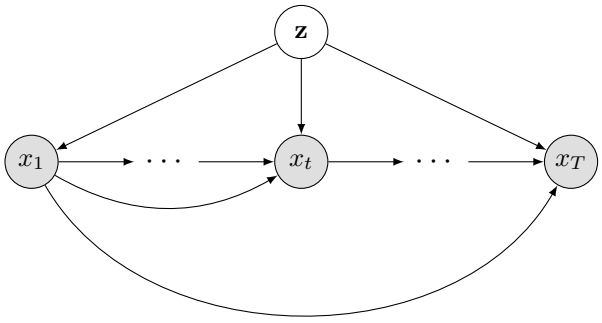


Figure 2.1 – A latent variable model with an observed variable with an autoregressive structure. This structure is seen in Gulrajani et al. (2016) and Chen et al. (2016). It is also the graphical model for Bowman et al. (2015), $p(\mathbf{x}|\mathbf{z}) = \prod_1^T p(x_t|x_1, \ldots, x_{t-1}, \mathbf{z})$

2.1 Related Work

From the modelling point of view, Chen et al. (2016) provides more background on why modelling some types of data in this manner is a good idea. If we consider a VAE that is trained to reconstruct the input entirely from the latent code, then the latent code is trained to reconstruct the entire input. However, not all the details of the given data point may be relevant. Gulrajani et al. (2016) and Chen et al. (2016) model images using auto-regressive decoders. These autoregressive decoders model the local variations (pixels close together on the image), while being conditioned on the latent code, which was responsible for higher level aspects of the image (general location, what was in the image etc.)

In the domain of language modelling, the first proposed PLVLM was Bowman et al. (2015), which introduced a sentence-level latent variable. The authors introduced the concept of *homotopies* between sentences. By interpolating between the latent code of different sentences, one could examine what the neighbourhood of a given sentence looked like by decoding it. While an autoencoder can in effect learn a latent representation space, the authors demonstrate that the transition between two sentences were not as smooth. This is one of the advantages of having a distribution over the latent space with a distribution: modelling the distribution over the latent space as a smooth one forces the model to learn such a mapping.

Since the initial approaches, there have been other work building upon using a latent variable for language generation. In Miao and Blunsom (2016) the latent space is another language model. The method attempts to use language as a latent variable in order to achieve unsupervised summarisation. However, the method is not trivial to train because the latent space is a large and discrete. Existing methods for backpropagating gradients through such latent variables have high variance (Williams, 1992).

Zhang et al. (2016); Rajeswar et al. (2017); Press et al. (2017); Yu et al. (2017) have also explored methods for natural language generation using Generative Adversarial Networks (GANs). While GANs have enjoyed great success in image generation due to their better image quality, they have not been shown to be advantageous in language generation. Moreover, there is the other caveat faced by GANs: it is difficult to evaluate and compare two different models / methods.

2.2 Implementation Details

2.2.1 Optimisation tricks

While VAEs are a popular technique for generative modelling, when using an auto-regressive decoder, the *latent variable collapse* problem (Dieng et al., 2018), also known as the *information preference problem* (Chen et al., 2016), is a common problem that arises.

Here, the decoder is a conditional language model, and RNN LMs are known to work well on their own. Consequently, the latent variable is often ignored, resulting in an unconditioned language model, and the model does not learn anything about the global context. There are several Strategies in existing literature to combat this.

β annealing

This technique augments the ELBO with a β multiplier at the KL-divergence term,

$$-\mathcal{L}(\theta, \phi; x) = -\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] + \beta \cdot D_{\mathrm{KL}}\left(q_\phi(\mathbf{z}|\mathbf{x}) \,\|p_\theta(\mathbf{z})\right) \tag{2.1}$$

Notice that when $\beta = 0$, the loss is simply reconstructing the input through a noisy channel. Since the objective of the encoder is only to provide a good representation for the decoder, the entropy of the resulting approximate posterior will be reduced. Reducing the penalty of the KL-divergence term initially during training will allow the decoder to use the latent code at the beginning, before slowly re-introducing the regularising penalty on the latent space. This means that, initially, the latent code is unconstrained and faces little penalty for encoding anywhere in the latent space. As β is gradually increased, the model is constrained to pack the latent code into the prior.

There are caveats to this method, however. Figure 2.2 shows the effects of annealing β at different rates. Ideally, different rates of β annealing should converge (eventually) at the same point. Unfortunately, a longer annealing schedule affects the reconstruction loss, and the entropy of the approximate posterior. Nevertheless, β-annealing remains a popular method for dealing with this issue.

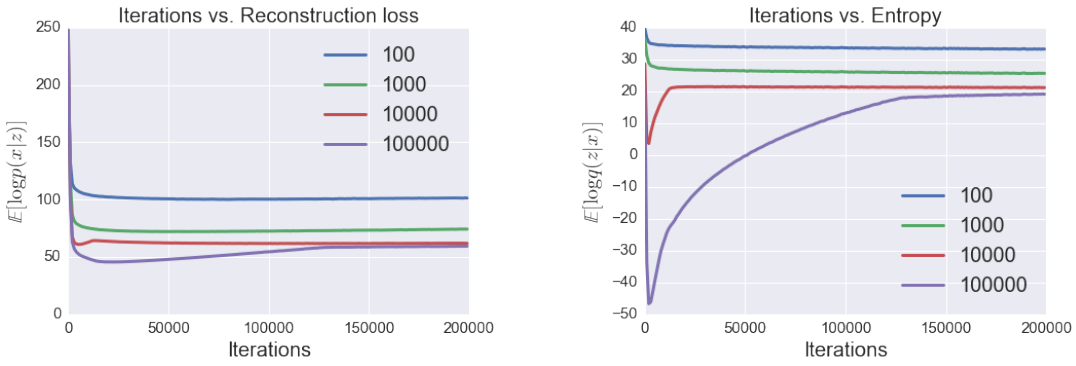


Figure 2.2 – The reconstruction loss and entropy over training iterations. The model is a VAE trained on the MNIST dataset.

Latent variable distribution

In Guu et al. (2017), the authors use a von Mises-Fisher (vMF) distribution instead of the more widely used Gaussian distribution for the latent variable. Due to its nature as a distribution of a point on a sphere, samples drawn from this distribution have a fixed magnitude. This along with their fixed parameterisation of the κ parameter causes the resulting KL-divergence term in the VLB have no regularisation effect on the latent variable. Davidson et al. (2018) later provides a more detailed treatment of the vMF distribution, along with a method for optimising κ.

Fundamentally, this method works similarly to β-annealing in that they both target the posterior distribution and the resulting regularising term in the ELBO that acts on it.

Reducing autoregressive context

An RNN decoder during learning predicts the data (x_t) based on all of the previous parts $(x_1, \ldots, x_{t-1})$. During training, the ground truth prior context is provided to the decoder. This is known as *teacher-forcing* and provides a lot of context for the decoder to predict the next word. Removing what the decoder is conditioned on during teacher-forcing forces the generative model to be further reliant on the latent code.

There are two methods to achieve this. In Bowman et al. (2015), the authors drop out certain words during decoding, replacing them with the `<unk>` symbol.

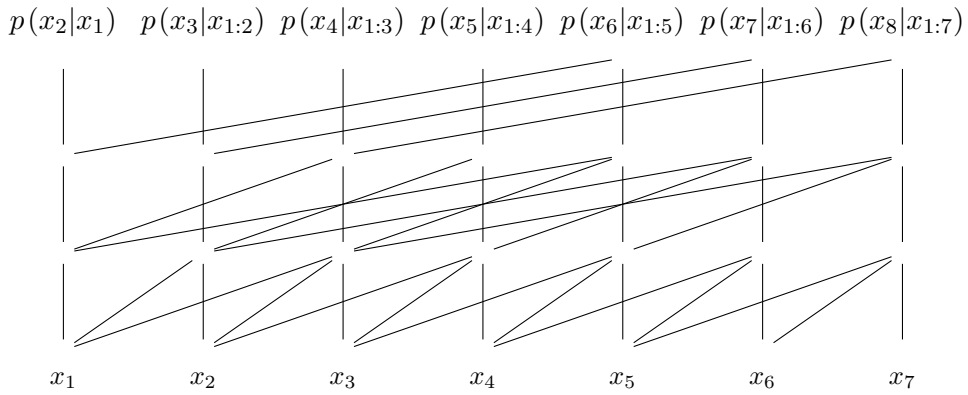


Figure 2.3 – An example of a 1-dimensional dilated convolution for language modeling.

Yang et al. (2017) uses a CNN to restrict the context size to a fixed-width context, in order to limit the capacity of the decoder.

Scaling Parameterisation

When conditioning the decoder on the latent variable $\mathbf{z}$, a straightforward parameterisation is to introduce it as an additional input to an MLP, or RNN. This means that the input layer is usually parameterised as follows:

$$\text{nonlinearity}\left(\mathbf{W}\mathbf{x} + \mathbf{U}\mathbf{z} + \mathbf{b}\right)$$

This method of parameterisation becomes problematic during the training process as $\mathbf{U} \to \mathbf{0}$. This is likely to happen as a result of not getting a useful signal from a noisy $\mathbf{z}$, which results in latent variable collapse. The problem is then compounded due to the higher variance.

Krueger et al. (2017) use a weight normalisation parameterisation (Salimans and Kingma, 2016). As the latent variable controls the scaling of the weights in the network, the connecting weights cannot collapse to zero or the model will fail to make any prediction.

2.2.2 Comparable Language Model Evaluation

The sentence-level latent variable model is handicapped under the standard evaluation of language models. A language model is usually evaluated by its

performance over the entire dataset, and, accordingly, the training of these models also take this into account. We need to make several small adjustments to the model presented in Bowman et al. (2015), and also the evaluation of $\log p(\mathbf{x})$ if we want to make the results comparable.

If we were to work under the same regime as existing state-of-the-art language models, then each word is always conditioned on the preceding context, disregarding sentence boundaries. At the sentence level, we can achieve something similar by conditioning each sentence on the previous sentence. However, in a latent variable model, this means sentences are conditioned on the past latent variables variables as well.

$$
\begin{aligned}
&\int \prod_{n}^{N} p\left(\mathbf{X}_n | \mathbf{z}_n, \mathbf{X}_{1:n-1}, \mathbf{z}_{1:n-1}\right) p\left(\mathbf{z}_n\right) \mathrm{d}\mathbf{z}_{1:N} \\
&= \mathbb{E}_{q(\mathbf{z}_{1:N}|\mathbf{X}_{1:N})}\left[\frac{1}{q\left(\mathbf{z}_{1:N}|\mathbf{X}_{1:N}\right)} \prod_{n}^{N} p\left(\mathbf{X}_n | \mathbf{z}_n, \mathbf{X}_{1:n-1}, \mathbf{z}_{1:n-1}\right) p\left(\mathbf{z}_n\right)\right] \\
&\approx \frac{1}{K} \sum_{k=1}^{K} \frac{1}{q\left(\mathbf{z}_{1:N}^{(k)} \middle| \mathbf{X}_{1:N}\right)} \prod_{n}^{N} p\left(\mathbf{X}_n \middle| \mathbf{z}_n^{(k)}, \mathbf{X}_{1:n-1}, \mathbf{z}_{1:n-1}\right) p\left(\mathbf{z}_n^{(k)}\right) \\
&\qquad\qquad \mathbf{z}_{1:N}^{(k)} \sim q\left(\mathbf{z}_{1:N}|\mathbf{X}_{1:N}\right)
\end{aligned}
$$

Therefore, in order to evaluate the model's likelihood in this setting, we would need to run K instances of each model across the entire dataset. This is because the sentence n is conditioned on all preceding sentences $\mathbf{X}_{1:n-1}$ and their sampled latent variable $\mathbf{z}_{1:n-1}$.

Armed with this, we can evaluate latent variable models at the dataset-level just like current benchmarks on language models.

2.3 Benchmarking Models on Penn Treebank

We benchmark existing architectures on the Penn Treebank dataset (PTB) for language modelling. All models are based off the PyTorch (Paszke et al.,

2017) example codebase for RNN language models [1]. As not all of the papers have benchmarked their method on PTB, we implement these models and their non-latent variable counterparts.

2.3.1 Experimental details

We use the larger learning rates advocated by the PyTorch example dataset and used averaged SGD (ASGD) or Polyak Averaging (Polyak and Juditsky, 1992) at the recommendation of Merity et al. (2017). We have also tried to reproduce both a latent variable dilated convolution language model, and one *without* a latent variable.

We use a learning rate of 20 based on Merity et al. (2017), and decay the learning rate by 0.25 every time the validation set error increases from the previous best. Once the learning rate reaches 3e-4, we switch to the Adam (Kingma and Ba, 2014) optimizer. This is run for 500 epochs and the best performance on the validation set is saved. We perform a hyper-parameter search over the dropout rate, embedding size and learning rate for the RNNLM model and the VAE models. We use the same hyper-parameters for the non-VAE models for comparison.

Decoder

We experiment with two types of decoders, an LSTM-RNN language model, and a dilated convolution model as described in Yang et al. (2017). All models share the embedding layer and the output layer (pre-softmax).

— **RNNLM** 2-layer LSTM language model with a hidden state and embedding size of 650.

— **Dilated Convolutions** 4-layer dilated 1D convolutions. The dilations at each layer is given by $d_i = 2^{l-1}$. The context window at each word is the 30 previous words.

Conditional Scale and Bias We also modify the final layer before the softmax to include a scale and bias conditioned on the latent variable with the following

1. `https://github.com/pytorch/examples/tree/master/word_language_model`

formulation:

$$\mathbf{s} = \mathbf{W}_s \mathbf{z} \qquad \mathbf{b} = \mathbf{W}_b \mathbf{z} \tag{2.2}$$

$$\mathbf{h}_t = \tanh\left(\frac{\widetilde{\mathbf{h}}_t - \mathbf{b}}{\mathbf{s} + \varepsilon}\right) \tag{2.3}$$

This method is to alleviate problems that we have mentioned in Section 2.2.1.

Approximate Posterior

We parameterise the approximate posterior using a 2-layer BiLSTM over the sentence, each with 650 hidden units. We then perform the following transforms on the output of the BiLSTM:

$$(\mathbf{h}_1, \ldots, \mathbf{h}_T) = \text{BiLSTM}(\mathbf{x}_1, \ldots, \mathbf{x}_T) \tag{2.4}$$

$$\widetilde{\mathbf{h}}_t = \mathbf{W}_{\text{pool}} \mathbf{h}_t \tag{2.5}$$

$$h_i^{\max} = \max_t \widetilde{h}_{t,i} \qquad h_i^{\text{mean}} = \frac{1}{T} \sum_t \widetilde{h}_{t,i} \tag{2.6}$$

$$\mathbf{c} = \text{MLP}\left(\mathbf{h}^{\max}, \mathbf{h}^{\text{mean}}\right) \tag{2.7}$$

We also experiment with using a conditional Gaussian for the approximate posterior and normalising flows, namely, a sequence of IAF transformations (Kingma et al., 2016). We use a 3-layer autoregressive MLP for each IAF transform, each layer is expanded to 8 times the original dimension. We use 4 such transformations, permuting the dimensions at each transform.

MADE Prior

We also run an experiment using a learned prior. For this, we use a MADE (Germain et al., 2015) model for the prior. However, instead of a randomised mask, we used an ordered mask with a block size of 8.

2.3.2 Results

It should be noted that these results are preliminary and there has not been an extensive hyperparameter search for each of these scenarios.

Table 2.1 – Results for language model experiments. For both sentence-level and full-corpus evaluation, we breakdown the *test* perplexity and the ELBO breakdown on the *train* set.

	Sentence		**Full**	
	Ppl.	Recon. + KL (Nats)	Ppl.	Recon. + KL (Nats)
RNN	87	-	77	-
+ VAE (Ours)	88	∼4.6 + 0.09	83	∼4.3 + 0.07
Bowman et al. (2015)	119	-	-	-
Dilated Convolution	90	-	89	-
+ VAE	88	∼4.2 + 0.22	94	∼4.0 + 0.36
+ MADE prior	86	∼4.0 + 0.34	92	∼3.9 + 0.33

Our sentence-level results are much better than what was reported in Bowman et al. (2015). This is likely due to the better baseline performance we attain from the RNN LM training setup that was used. However, like Bowman et al. (2015), the VAE model performed worse than the standard RNN LM model did. Our dilated convolution LM models are smaller than the ones applied in Yang et al. (2017) because our initial hyper-parameter search was performed on the sentence-level modelling task. Since the average length of the PTB training set is 21.1 words, it is unsurprising that the optimal number of layers would be 4, which results in a context size of 31. This would allow the entire sentence to be within the context for the last words being predicted. Interestingly, using the MADE prior gave us results for sentence modelling that are much better than our experiment using an RNN. However, it should be noted that the RNN sentence-level model setup was based on the **Full** hyperparameter setup, and that a more thorough hyperparameter search might yield different results.

As mentioned in Section 2.2.2, the evaluation method on the **Full** set up is different. Instead of taking the mean over the samples for each sentence, the mean is taken over the final joint probability. This may explain why the results for the full evaluation are poorer when the decoder depends more on the latent variable.

Samples

Conditioned on fixed latent Most of the samples we attained from the models when fixing a particular latent variable sample showed the same initial prefix. This suggests that the latent variable models a lot of the initial words, which is not surprising, as the highest uncertainty when reconstructing a sentence independent

```
the dow jones industrials closed up N to N <eos>
the dow jones industrials closed up N points to N <eos>
the dow jones industrials closed up N points to N <eos>

the market 's <unk> is that the market is still <unk> <eos>
but the market 's <unk> is a <unk> <eos>
the market 's <unk> is that the market is still <unk> <eos>

in tokyo the nikkei index of N points to N <eos>
in tokyo the nikkei index of N points to N <eos>
in tokyo the nikkei index of N points to N <eos>

the issue was priced at N to yield N N <eos>
the issue was priced at N to yield N N <eos>
the issue was priced at N to yield N N <eos>
```

Figure 2.4 – Groups of sentence samples given the same sample from the prior.

```
the problem has n't been <unk> <eos>

the company is n't extremely difficult to have a good impact
on the industry says mr. <unk> <eos>

mr. <unk> said the company is n't sure that the company 's
stock has been growing that its <unk> is n't <unk> <eos>

he said the company is n't going to have a look at the <unk> <eos>

but there are no question that the market is n't being acquired <eos>

the company is that the company 's earnings have been <unk> <eos>

the company 's <unk> is n't the result of a <unk> <eos>

the company has been a very soft <unk> <eos>

in particular the company is n't profitable says <unk> <unk>
an analyst with drexel burnham lambert inc <eos>

the company said it does n't have any bearing on the company 's
financial problems <eos>

the company is n't as difficult as the company 's
financial troubles <eos>
```

Figure 2.5 – Interpolating two latent variables samples from the prior.

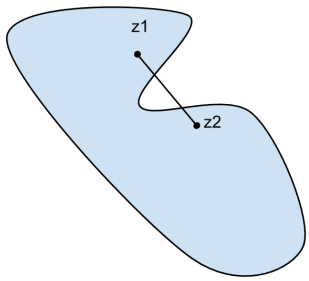


Figure 2.6 – Interpolating in an irregular distribution may resulting in traversing past low probability coordinates.

of any additional information are the initial words.

Interpolating We sample two points from the learned prior, $\mathbf{z}_1$ and $\mathbf{z}_2$, and sampled points along $\mathbf{z}_1 + \frac{i \cdot \mathbf{z}_2}{n}$ for $i \in (0, \ldots, n)$. In Figure 2.5, we use $n = 10$ and sample from the decoder. Interestingly, the sentences all have relative negative sentiment associated with them, with words like "`has 'nt`", "`is 'nt`", "`does 'nt`" repeatedly coming up. Aside from the first sentence, all of them mention "`company`" The same phenomenon with the groups of samples conditioned on the same latent variable is present: the first few words are common among the samples.

It is worth noting that interpolating linearly in the latent space may not be the right thing to do if the prior can be learned (See Figure 2.6). Under a standard Gaussian prior, any interpolated point between two sampled points would have more density than one of the points. With a learned prior, there is no guarantee that there is much density in between the two sample points.

2D Latent Space We also trained a two dimensional latent space for the dilated convolution model. However, the KL-divergence in models trained with 2D latent spaces were consistently low. We suspect this is due to the two dimensions being too small a bottleneck to provide any useful information to the decoder.

Figure 2.7 shows the difference in the learned space. We sampled three different sentences from the validation set, and a 100 latent samples from $\mathbf{z}$-space. The Gaussian approximate posterior (Vanilla VAE) has a variance that is aligned with the axes, while the one with IAF is aligned diagonally. While IAF could have

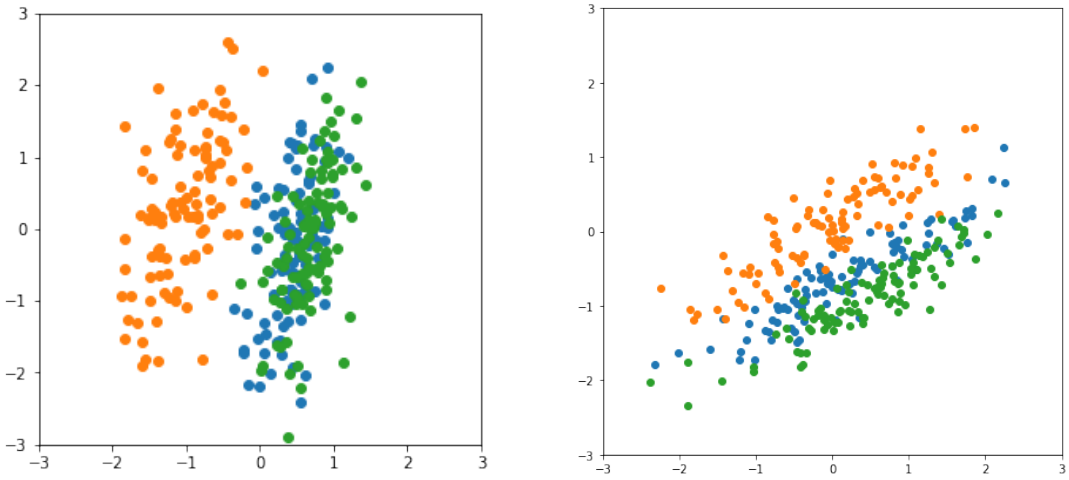


Figure 2.7 – Left: 2D latent space with a conditional Gaussian approximate posterior. Right: 2D latent space with IAF approximate posterior.

modelled a multi-modal posterior distribution, but this did not happen in our case. The KL term in the ELBO was also much lower in comparison to the 100 dimenson latent variable.

2.3.3 Discussion & Future Work

While these are preliminary results, we believe that the approach is promising and should not be abandoned. Conceptually, PLVLMs should attain at least similar results when compared to standard RNNLM. The latent variable collapse problem is indeed an issue that should be treated with care, and thinking about them in terms of the prior, approximate posterior and the decoder is useful to ameliorating the issue. Specifically, in our case:

— The **decoder** had a lot of modelling capability. This was reduced by giving it a smaller context with the dilated convolution. We also modified the decoder so that the architecture required using the latent variable to some extent.

— We experimented with using an IAF **approximate posterior**. However, the results were comparable to that of using the standard Gaussian.

— We used a MADE **prior** for the dilated convolution set up, and found that this improved the results, suggesting that a learnable prior may be crucial for a latent variable language model.

However, more subjectively, the samples from the model showed issues with this

method of modelling. The latent variable seemed to be largely modelling the initial parts of the sentence, with phrases like "`the company`" often showing up. This is perhaps to be expected, and an objective test of the learned latent variables should be done before drawing any conclusions. This should include applying the inferred latent representations on a suite of tasks like GLUE

More pressing follow-up work that should be done is a more comprehensive suite of experiments on different combinations of set ups: RNNs vs. CNNs, normalising flows vs. conditional Gaussians, learned prior vs. standard Gaussian prior, etc.

Along the same lines, other interesting avenues of exploration include:

Conditional Priors One other way to improve things is to condition the prior on previous sentence representations. Serban et al. (2017) uses an RNN on the sentence level representations. This could be extended to probabilistic latent variables where there would be a distribution over the higher level latent variables.

Information Bottleneck between Words We may be able to take the variational view of dropout from Gal and Ghahramani (2016) and apply that to the recurrent function parameters.

Taking the information bottleneck perspective again, one possible way to overcome the problem would be to apply an information bottleneck at the RNN transition. In the context of variational learning, this translates to making the hidden state of the model a latent variable.

Sohl-Dickstein and Kingma (2015) also looks at the hidden states of the RNN ($\mathbf{h}_t$) as a latent variable, and the standard RNN is a specific case where the distribution over $\mathbf{h}_t$ is a Dirac delta. The authors then consider distributions over $\mathbf{h}_t$ that are not Diracs. As a consequence of this, we could have multiple particles over the sequence, giving multiple different trajectories. This may result in a lower dimensional hidden state with a multi-modal distribution, instead of a Dirac over a large hidden state to represent the context. Figure 2.8 illustrates the graphical model associated with this.

In comparison to the other methods, this set up could potentially learn to balance the global variable usage $\mathbf{w}$ with the per-word latent usagel $\mathbf{z}_t$. Edwards and Storkey (2016) has a similar structure, but did not apply their model on natural

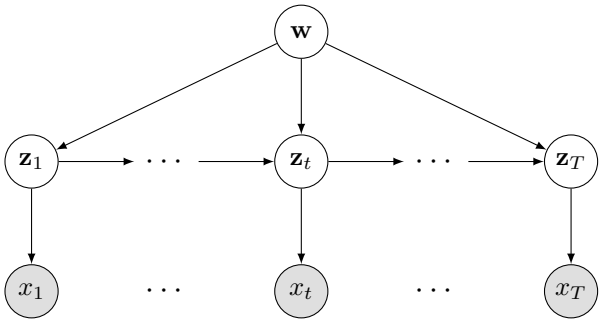


Figure 2.8 – Hidden states of the RNN as stochastic latent variables, as discussed in Sohl-Dickstein and Kingma (2015).

language.

Merity et al. (2017) reported huge differences in their ablation study when DropConnect (Wan et al., 2013) on the transition function was removed from their RNN LM. However, from our own empirical studies, randomly dropping out different hidden states at each time-step harms performance. This suggests that there are specific types of noise are crucial to learning a good language model, and when applied at the right place in the model, it can drastically improve results.

Another view one can take of the conditional distribution over the latent space is in ambiguity about the meaning of the sentence. In the next chapter, we use an Adversarial Autoencoder (AAE) setup to learn a conditional distribution over the latent space in the hopes of learning such a distribution.

3 Generating Entailments, Contradictory and Neutral Sentences

3.1 Prologue

Shen et al. (2018) **Unpublished**

Personal Contribution The original conception of the idea was to come up with a method for performing logical operations on different sentences, analogous to the way it is possible to do vector arithmetic using word embeddings. This was with the final goal of performing multi-document / multi-sentence aggregation: merge multiple documents / sentences into one which contains all the available information in the given sentences.

Huang Chinwei initially proposed the idea of thinking of logical relationships between sentences as operators on an original sentence. The initial model prototype was coded up by Shen Yikang, while I conceived of methods for evaluating the samples from the model. I prepared most of the evaluation metrics for the paper, particularly the SotA Stanford Natural Language Inference (SNLI) models.

There was also an attempt to generate sentences with combined facts from two sentences by searching in the latent space through gradient descent, but because it was not successful, was not included in the paper.

3.2 Introduction

Algorithms designed to learn distributed sentence representations have been shown to be transferable across a range of tasks (Mou et al., 2016) and languages (Tiedemann, 2018). For example, Guu et al. (2017) proposed to represent sentences as vectors that encode a notion similarity between sentence pairs, and showed that vector manipulations of the representation can result in meaningful change in semantics. The question we would like to explore is whether the semantic

relationship between sentence pairs can be modeled in a more explicit manner. More specifically, we want to model the *logical relationship* between sentences.

Controlling the logical relationship between sentences has many direct applications. First of all, we can use it to provide a more clear definition of paraphrasing. To do so, we require two simultaneous conditions: (i) that the input sentence *entails* the output sentence; and (ii) that the output sentence *entails* the input sentence.

$$\begin{aligned}(\textsc{Sentence1} \models \textsc{Sentence2}) \wedge \\ (\textsc{Sentence2} \models \textsc{Sentence1})\end{aligned} \tag{3.1}$$

The first requirement ensures the output sentence cannot be false if the input sentence is true, so that the output sentence can be considered a fact expressed by the input sentence. The second requirement ensures that the output contains at least the input's information. The two requirements together can be used to define semantic equivalence between sentence.

Another interesting application is multi-document summarization. Traditionally, to summarize multiple documents, one would expect the model to abstract the most important part of the source documents, and this is usually measured by the amount of overlap that the output document has with the inputs. Informally, one finds the maximal amount of information that has the highest precision with each source document. Alternatively, if one wants to automate news aggregation, the ideal summary would need to contain the same number of facts as are contained in the union of all source documents. We can think of this second objective as requiring that the output document entail every single sentence across all source documents.

In this paper, we propose an approach to generating sentences, conditioned on an input sentence and a logical inference label. We do this by modeling the different possibilities for the output sentence as a distribution over the latent representation, which we train using an adversarial objective.

In particular, we differ from the usual adversarial training on text by using a differentiable global representation. Architecture-wise, we also propose a Memory Operation Selection Module (MOSM) for encoding a sentence into a vector representation. Finally, we evaluate the quality and the diversity of our samples.

3.3 Related Work

Many natural language tasks require reasoning capabiliities. The Recognising Textual Entailment (RTE) task requires the system to determine if the *premise* and *hypothesis* pair are (i) an entailment, (ii) contradicting each other or (iii) neutral to each other. The Natural language Inference (NLI) Task from Bowman et al. (2015) introduces a large dataset with labeled pairs of sentences and their corresponding logical relationship. This dataset allows us to quantify how well current systems are able to be trained to recognise sentences with those relationships. Examples of the current state-of-the-art for this task include Chen et al. (2017) and Gong et al. (2017).

Here we are interested in generating natural language that satisfies the given textual entailment class. Kolesnyk et al. (2016) has attempted this using only sentences from the entailment class, and focusing on generating a hypothesis given the premise. Going in this direction results in removal of information from the premise sentence. In this paper, we focus on going in the other direction: generating a premise from a hypothesis. This requires adding additional details to the premise which have to make sense in context. In order to produce sentences with extra details and without some other details, we suggest that a natural way to model this kind of structure is to impose a distribution over an intermediate distribution representing the semantic space of the premise sentence.

In the realm of learning representations for sentences, Kiros et al. (2015) has a popular method for learning representations called "skip-thought" vectors. These are trained by using the encoded sentence to predict the previous and next sentence in a passage. Conneau et al. (2017) specifically learned sentence representations from the SNLI dataset. They claim that using the supervised data from SNLI can outperform "skip-thought" representations on different tasks. There have also been several efforts towards learning a distribution over sentence embeddings. Bowman et al. (2015) used Variational Autoencoders (VAEs) to learn Gaussian distributed word embeddings. Hu et al. (2017) use a combined VAE/GAN objective to produce a disentangled representation that can be used to modify some attributes like sentiment and tense.

There have also been forays into conditional distributions for sentences – which is what is required here. Both Gupta et al. (2017) and Guu et al. (2017) introduce

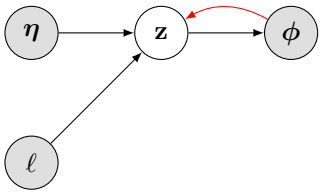


Figure 3.1 – The conceptual graphical model behind the formulation of our model. The red arrow represents the inference path from ϕ to $\mathbf{z}$.

models of the form $p(\mathbf{x}|\mathbf{z}, \mathbf{x}')$, where $\mathbf{x}$ is a paraphrase of $\mathbf{x}'$, and $\mathbf{z}$ represents the variability in the output sentence. Guu et al. (2017) introduces $\mathbf{z}$ as an edit vector. However, because $\mathbf{z}$ has to be paired with $\mathbf{x}'$ in order to generate the sentence, $\mathbf{z}$ serves a very different purpose, and cannot be considered a sentence embedding in its own right. Ideally, what we want is a distribution over sentence representations, each one mapping to a set of semantically similar sentences. This is important if we want the distribution to model the possibilities of concepts that correspond to the right textual entailment with the hypothesis.

3.4 Method

Some approaches map a sentence to a distribution in the embedding space (Bowman et al., 2015). The assumption when doing this is that there is some uncertainty over the latent space when mapping from the sentence. Some approaches, like Hu et al. (2017) attempt to disentangle factors in the learnt latent variable space, so that modifying each dimension in the latent representation modifies a factor, like sentiment or tense, in the original sentence.

If we consider plausible premise sentences $\boldsymbol{\phi}$ given a hypothesis $\boldsymbol{\eta}$ and an inference label ℓ, there are many possible solutions, of varying likelihoods. We can model this probabilistically as $p(\boldsymbol{\phi}|\boldsymbol{\eta}, \ell)$. In our model, we assume an underlying latent variable $\mathbf{z}$ that accounts for the variation in possible output sentences,

$$p(\boldsymbol{\phi}|\boldsymbol{\eta}, \ell) = \int p(\boldsymbol{\phi}|\mathbf{z})p(\mathbf{z}|\boldsymbol{\eta}, \ell)\mathrm{d}\mathbf{z}$$

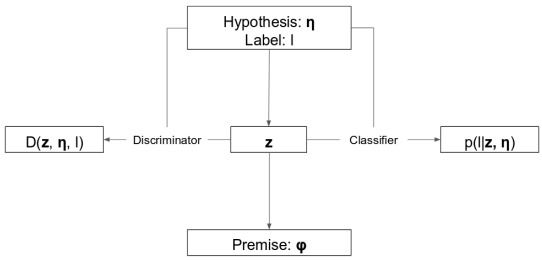


Figure 3.2 – The architecture of the model. The autoencoder maps given premise $\boldsymbol{\phi}$ to a sentence representation $\mathbf{z}$, and reconstructs $\boldsymbol{\phi}$ from $\mathbf{z}$. Samples are drawn from the prior conditioned on $\boldsymbol{\eta}$ and ℓ. The classifier takes $\mathbf{z}$ and $\boldsymbol{\eta}$ as input, and outputs probability of l. The discriminator takes $\mathbf{z}$, $\boldsymbol{\eta}$ and l as input, and predicts whether $\mathbf{z}$ is given by the autoencoder or the prior.

Another assumption we make is that given $\boldsymbol{\phi}$, $\mathbf{z}$ is independent of $\boldsymbol{\eta}$ and ℓ. The resulting graphical model associated with the above dependency assumptions are depicted in Figure 3.1.

In our proposed model, we take inspiration from the Adversarial Autoencoder (Makhzani et al., 2015). However, our prior is conditioned on $\boldsymbol{\eta}$ and ℓ. Zhang et al. (2017) also proposed a Conditional Adversarial Autoencoder for age progression prediction. In addition to the adversarial discriminator, our model includes a classifier on the representation and the hypothesis and label. A similar framework is also discussed in Salimans et al. (2016).

3.4.1 Architecture

The model consists of an encoder $q(\mathbf{z}|\boldsymbol{\phi})$, a conditional prior, $p(\mathbf{z}|\boldsymbol{\eta}, \ell)$, a decoder $p(\boldsymbol{\phi}|\mathbf{z})$, and a discriminator $\mathrm{D}(\mathbf{z}, \boldsymbol{\eta}, \ell)$.

Autoencoder The autoencoder comprises of two parts. An encoder that maps the given premise $\boldsymbol{\phi}$ to a sentence representation $\mathbf{z}$, and a decoder that reconstructs $\boldsymbol{\phi}$ from a given $\mathbf{z}$. In our model, the encoder reads the input premise $\boldsymbol{\phi} = (x_1^{\boldsymbol{\phi}}, ..., x_{|\boldsymbol{\phi}|}^{\boldsymbol{\phi}})$ using an RNN network:

$$h_1^{\boldsymbol{\phi}}, ..., h_{|\boldsymbol{\phi}|}^{\boldsymbol{\phi}} = \mathrm{RNN}_{\mathrm{enc}}(x_1^{\boldsymbol{\phi}}, ..., x_{|\boldsymbol{\phi}|}^{\boldsymbol{\phi}}) \tag{3.2}$$

and

$$\mathbf{z} = f_{\mathrm{compress}}(h_1^{\boldsymbol{\phi}}, ..., h_{|\boldsymbol{\phi}|}^{\boldsymbol{\phi}}) \tag{3.3}$$

where $h_t \in \mathcal{R}^n$ is a hidden state at time t. $\mathbf{z}$ is a vector computed from the sequence of the hidden states. We will call $f_{\text{compress}}(\cdot)$ the compression function.

The decoder is trained to predict the next word x'_t given the sentence representation $\mathbf{z}$ and all the previously predicted words $(x'_1, ..., x'_{t-1})$. With an RNN, the conditional probability distribution of x'_t is modeled as:

$$s_1, ..., s_{t-1} = \text{RNN}_{\text{dec}}(x'_1, ..., x'_{t-1}) \tag{3.4}$$

$$c_{t-1} = f_{\text{retrieve}}(\mathbf{z}, s_{t-1}) \tag{3.5}$$

so,

$$p(x'_t | x'_1, ..., x'_{t-1}, \mathbf{z}) = g(s_{t-1}, c_{t-1}) \tag{3.6}$$

where $g(\cdot)$ is a nonlinear, potentially multi-layered, function that outputs the probability of x'_t, s_t is the hidden state of decoder RNN, and $f_{\text{retrieval}}$ takes s_t as the key to retrieve related information from $\mathbf{z}$. We note that other architectures such as a CNN or a transformer Vaswani et al. (2017) can be used in place of the RNN. The details of the compression function and retrieval function will be discussed in Sec. 3.4.2.

Prior We draw a sample, conditioned on $(\boldsymbol{\eta}, \ell)$, through the prior, which is described using following equations:

$$h_1^{\boldsymbol{\eta}}, ..., h_{|\boldsymbol{\eta}|}^{\boldsymbol{\eta}} = \text{RNN}_{\text{enc}}(x_1^{\boldsymbol{\eta}}, ..., x_{|\boldsymbol{\eta}|}^{\boldsymbol{\eta}}) \tag{3.7}$$

$$\tilde{h}_t = \text{MLP}([h_t^{\boldsymbol{\eta}}, e_\ell, \varepsilon]) \tag{3.8}$$

$$\hat{h}_1, ..., \hat{h}_{|\boldsymbol{\eta}|} = \text{RNN}_{\text{refine}}(\tilde{h}_1, ..., \tilde{h}_{|\boldsymbol{\eta}|}) \tag{3.9}$$

$$\mathbf{z} = f_{\text{compress}}(\hat{h}_1, ..., \hat{h}_{|\boldsymbol{\eta}|}) \tag{3.10}$$

where ε is a random vector, $\varepsilon_i \sim \mathcal{N}(0, 1)$; e_ℓ is the label embedding and $[\cdot, \cdot]$ represents the concatenation of input vectors.

Classifier This outputs the probability distribution over labels, taking as input the tuple $(\mathbf{z}, \boldsymbol{\eta})$, and is described using the following equations:

$$
\begin{aligned}
h_1^{\boldsymbol{\eta}}, ..., h_{|\boldsymbol{\eta}|}^{\boldsymbol{\eta}} &= \text{RNN}_{\text{enc}}(x_1^{\boldsymbol{\eta}}, ..., x_{|\boldsymbol{\eta}|}^{\boldsymbol{\eta}}) && (3.11)\\
c_t &= f_{\text{retrieve}}(\mathbf{z}, h_t^{\boldsymbol{\eta}}) && (3.12)\\
\tilde{h}_t &= \text{MLP}([h_t^{\boldsymbol{\eta}}, c_t, ||h_t^{\boldsymbol{\eta}} - c_t||, h_t^{\boldsymbol{\eta}} \odot c_t]) && (3.13)\\
\hat{h}_1, ..., \hat{h}_{|\boldsymbol{\eta}|} &= \text{RNN}_{\text{refine}}(\tilde{h}_1, ..., \tilde{h}_{|\boldsymbol{\eta}|}) && (3.14)\\
\hat{h}_{\text{max}} &= \text{Pooling}_{\text{max}}(\hat{h}_1, ..., \hat{h}_{|\boldsymbol{\eta}|}) && (3.15)\\
\hat{h}_{\text{mean}} &= \text{Pooling}_{\text{mean}}(\hat{h}_1, ..., \hat{h}_{|\boldsymbol{\eta}|}) && (3.16)\\
p(\ell|\mathbf{z}, \boldsymbol{\eta}) &= \sigma(\text{MLP}([\hat{h}_{\text{max}}, \hat{h}_{\text{mean}}])) && (3.17)
\end{aligned}
$$

where Pooling$(\cdot)$ refers to an element-wise pooling operator, and the activation function σ for output layer is the softmax function. The architecture of the classifier is inspired by (Chen et al., 2017). Instead of doing attention over the sequence of hidden states for the premise, we use the retrieval function in Equation 3.12 to retrieve related information c_t in $\mathbf{z}$ for $h_t^{\boldsymbol{\eta}}$.

Discriminator The discriminator takes as input $(\mathbf{z}, \boldsymbol{\eta}, \ell)$, and tries to determine if the $\mathbf{z}$ in question comes from the encoder or prior. The architecture of the discriminator is similar to that of the classifier, with the exception that Equation 3.13 is replaced by:

$$\tilde{h}_t = \text{MLP}([h_t^{\boldsymbol{\eta}}, c_t, e_\ell]) \quad (3.18)$$

to pass label information to the discriminator. The sigmoid function is used as the activation for the output layer.

In our model the autoencoder, prior and classifier share the same $\text{RNN}_{\text{enc}}(\cdot)$ parameters. The prior and the autoencoder share the same $f_{\text{compress}}(\cdot)$ parameters. The classifier and the autoencoder share the same $f_{\text{retrieve}}(\cdot)$ parameters. The discriminator does not share any parameters with the rest of model.

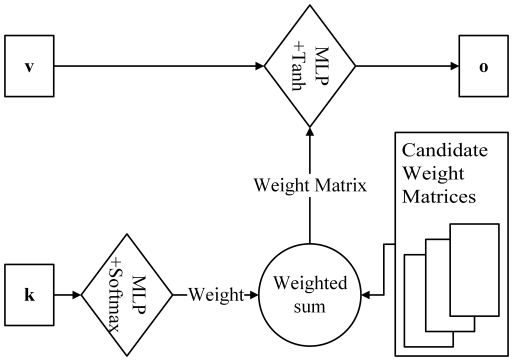


Figure 3.3 – *Memory Operation Selection Module* takes a pair of vector $(\mathbf{k}, \mathbf{v})$ as input, output a vector $\mathbf{o}$. $\mathbf{k}$ provide the control signal for the layer to compute a weighted sum of candidate weight matrices. The obtained matrix is used as the weight matrix in a normal feedforward layer, that takes $\mathbf{v}$ as input and outputs $\mathbf{o}$.

3.4.2 Compression and Retrieval Functions

The compression (Equation 3.3) and retrieval (Equation 3.5) functions can be modeled through many different mechanisms. Here, we introduce two different methods:

Mean Pooling can be used to compress the sequence of the hidden states:

$$f_{\text{compress}}(h_1, ..., h_T) = \frac{1}{T} \sum_{t=1}^{T} h_t \tag{3.19}$$

and its retrieve counterpart directly returns $\mathbf{z}$:

$$f_{\text{retrieve}}(\mathbf{z}, s_t) = \mathbf{z} \tag{3.20}$$

Memory Operation Selection Module (MOSM) As an alternative to mean pooling, we use the architecture shown in Figure 3.3. A layer is defined as:

$$\gamma = \text{softmax}(\mathbf{\Omega k}) \tag{3.21}$$

$$\tilde{\mathbf{W}} = \sum_{i=1}^{N_{\mathbf{W}}} \gamma_i \mathbf{W}_i \tag{3.22}$$

$$\mathbf{o} = \sigma(\tilde{\mathbf{W}} \mathbf{v}) \tag{3.23}$$

where σ can be any activation function, $\mathbf{v}$ is the input vector, $\mathbf{k}$ is the control vector, $\{\mathbf{W}_i\}$ are $N_{\mathbf{W}}$ candidate weight matrices. For convenience, we denote the MOSM function as $f_{\text{MOSM}}(\mathbf{v}, \mathbf{k})$.

Thus, we can define the MOSM compression method as:

$$f_{\text{compress}}(h_1, ..., h_T) = \tanh\left(\frac{1}{T}\sum_{t=1}^{T} f_{\text{MOSM}}\left(h_t, h_t\right)\right) \tag{3.24}$$

The compression function uses $\{h_t\}$ as both control and input vector, to write themselves into $\mathbf{z}$. Because different h_ts select different combinations of candidate matrices, we can have different mapping function each different h_t at each time step.

$$f_{\text{retrieve}}(\mathbf{z}, s_t) = f_{\text{MOSM}}(\mathbf{z}, s_t) \tag{3.25}$$

Retrieval functions use $\{s_t\}$ as control vectors to retrieve information from $\mathbf{z}$. Since the layer generates a different weight matrix for the feedforward path for different s_t, we can output different $\mathbf{o}$ for the same $\mathbf{z}$.

3.4.3 Model Learning

Like most adversarial networks, the conditional adversarial autoencoder is trained with a gradient descent based method in two phases: the *generative* phase and the *discriminative* phase.

In the *generative* phase, the autoencoder is updated to minimize the reconstruction error of the premise. The classifier and the encoder are updated to minimize the classification error of the premise-hypothesis pair. The prior is also updated to optimize the classification error of $p(\ell|\mathbf{z}, \boldsymbol{\eta})$, where $\mathbf{z}$ is draw from the prior. The encoder and the prior are updated to confuse the discriminator.

In our initial experiments, we found that the samples from just the adversarial training alone results in wildly varied output sentences. To ameliorate this, we propose an *auxiliary loss*:

$$\mathcal{L}_{\text{auxiliary}} = \min_{i \in (1,...,N)} \{\text{NLL}(\boldsymbol{\phi}|\mathbf{z}_i)\}\ , \quad \mathbf{z}_i \sim p(\mathbf{z}|\boldsymbol{\eta}, \ell) \tag{3.26}$$

where N is the number of samples that are drawn from prior. The auxiliary loss measures how far our generated premises are from the true premise when

conditioned on the hypothesis and label. As shown in experiment the model has better generating diversity, while more samples were drawn during training.

One can view this auxiliary loss as a 'hard' version of taking the log average of the probability of N Monte-Carlo samples,

$$- \log \mathbb{E}_{p(\mathbf{z}|\boldsymbol{\eta},\ell)} \left[p(\boldsymbol{\phi}|\mathbf{z}) \right] \tag{3.27}$$

$$\approx - \log \frac{1}{N} \sum_{i}^{N} p(\boldsymbol{\phi}|\mathbf{z}_i), \qquad \mathbf{z}_i \sim p(\mathbf{z}|\boldsymbol{\eta}, \ell) \tag{3.28}$$

$$= - \log \frac{1}{N} - \log \sum_{i}^{N} \exp \log p(\boldsymbol{\phi}|\mathbf{z}_i) \tag{3.29}$$

$$\leq - \log \frac{1}{N} + \min_{i \in (1,\ldots,N)} - \log p(\boldsymbol{\phi}|\mathbf{z}_i) \tag{3.30}$$

Since $\log \frac{1}{N}$ is a constant, minimizing over the Equation 3.30 is the same as minimizing Equation 3.26.

In the *discriminative* phase, the discriminator is updated to tell apart the true $\mathbf{z}$ (generated using the prior) from the generated samples (given by autoencoder).

3.5 Experiments

We use the Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015) to train and evaluate our models. From our experiments, we want to determine two things. First, do the sentences produced by the model form the correct textual entailment class on which it was conditioned on? Second, is there diversity among the sentences that are generated?

3.5.1 Baseline Methods

For comparison, we use a normal RNN encoder-decoder as a baseline method. The model uses a bidirectional LSTM network as encoder. The encoder reads the

input hypothesis into a sequence of hidden states $\{h_t\}$:

$$h_1^{\boldsymbol{\eta}}, .., h_{|\boldsymbol{\eta}|}^{\boldsymbol{\eta}} = \mathrm{RNN}_{\mathrm{enc}}(x_1^{\boldsymbol{\eta}}, .., x_{|\boldsymbol{\eta}|}^{\boldsymbol{\eta}}) \tag{3.31}$$

$$z_{\boldsymbol{\eta}} = f_{\mathrm{compress}}(h_1^{\boldsymbol{\eta}}, .., h_{|\boldsymbol{\eta}|}^{\boldsymbol{\eta}}) \tag{3.32}$$

Where $f_{\mathrm{compress}}(\cdot)$ can be the mean method or an MOSM. The distributed representation of label e_ℓ and $z_{\boldsymbol{\eta}}$ are concatenated together to feed into an MLP, which gives the sentence representation $\mathbf{z}$:

$$\mathbf{z} = \mathrm{MLP}([z_{\boldsymbol{\eta}}, e_\ell]) \tag{3.33}$$

The decoder then computes the conditional probability distribution with equations:

$$p(x_t'|x_1', ..., x_{t-1}') = g([s_t, \mathbf{z}]) \tag{3.34}$$

$$s_t = \mathrm{RNN}_{\mathrm{dec}}(x_{t-1}', s_{t-1}) \tag{3.35}$$

Thus, the baseline model share a similar architecture with prior and decoder in our model, while the randomness been toke out.

3.5.2 Experiment Settings

For all models, $\mathrm{RNN}_{\mathrm{enc}}$ and $\mathrm{RNN}_{\mathrm{refine}}$ are 2-layers bi-directional LSTM (Schuster and Paliwal, 1997), $\mathrm{RNN}_{\mathrm{dec}}$ are 2-layers uni-directional LSTM. The dimension of hidden state, embeddings and latent representation $\mathbf{z}$ are 300. When training, optimization is performed with Adam using learning rate $lr = 0.001$, $\beta_1 = 0$, $\beta_2 = 0.999$ and $\sigma = 10^{-8}$. We carry out gradient clipping with maximum norm 1.0. We train each model for 30 epoch. For each iteration, we randomly choose to run the generative phase or discriminative phase with probability $0.5 : 0.5$. Since we didn't observe significant benefit from using Beam Search, all premises are generated using greedy search.

Table 3.1 – Classification accuracies for different state-of-the-art models on our samples. The row labeled RANDOM we randomly permuted the premises of the original test set and ran them through the classifiers to test for the models' reliance on just the hypothesis for classification.

MODEL	DIIN	ESIM
RANDOM	42.7%	41.1%
BASELINE (MEAN)	59.6%	59.6%
BASELINE (MOSM)	62.7%	62.6%
MOSM (N=1, -CLASSIFIER)	67.2%	67.3%
MOSM (-AUXILIARY LOSS)	63.2%	60.6%
MEAN (N=1)	64.4%	62.4%
MEAN (N=10)	64.3%	62.3%
MOSM (N=1)	76.1%	75.9%
MOSM (N=10)	72.6%	71.8%

3.5.3 Quality Evaluation

In order to evaluate the quality of the samples from our model, we trained two state-of-the-art models for SNLI: (1) Densely Interactive Inference Network (DIIN) [1] (Gong et al., 2017), (2) Enhanced Sequential Inference Model (ESIM) [2] (Chen et al., 2017).

In our experiments, we found that it is possible to achieve an accuracy of 68% on SNLI label prediction by training a classifier using *only* the hypothesis as input. This calls into question how much the classification models rely on just the hypothesis for performing its task. To investigate this phenomena further, we randomly permuted the premises of the original test set and passed these new (random) premise-hypothesis pairs to the classifiers. The results are shown in the row labelled RANDOM in Table 3.1. We were satisfied that at 42.7% and 41.1%, the classification models (both DIIN and ESIM) were not relying entirely on the hypothesis for prediction.

We sampled 9845 hypotheses from the test set, and produced $\boldsymbol{\phi}$ for each example with the given ℓ. The $(\boldsymbol{\eta}, \boldsymbol{\phi}, \ell)$ triplet was then passed to the classifiers and evaluated for accuracy. Both classification models perform at ∼88% accuracy, but, while they were not perfect, they provided a good probe for how well our models were generating the required sentences. Table 3.1 shows the accuracy of prediction on

1. https://github.com/YichenGong/Densely-Interactive-Inference-Network
2. https://github.com/lukecq1231/nli

Table 3.2 – The confusion matrix for the samples from the best model MOSM ($N = 1$)

Label \Pred.	Ent.	Neut.	Cont.
Entailment	67.8%	20.9%	11.4%
Neutral	6.6%	76.7%	16.7%
Contradiction	2.9%	12.8%	84.4%

the respective models. Both the DIIN and ESIM models give similar results.

Our results show that using the MOSM gives an improvement over just taking the mean. Using the adversarial training also results in some gains, which suggests that training the model with the 'awareness' of the distribution over the representation space results in better quality samples. Using the adversarial training in conjunction with the MOSM layer gives us the model with the best performance. We also performed ablation tests, removing certain components of the model from the training to see how it affects the quality of samples. The difference between our best model against MOSM ($N = 1$, -classifier) suggests that the classifier plays in important role in ensuring $\mathbf{z}$ is a representation in the right class. In our experiment removing the auxiliary loss, we still achieve an accuracy ∼61%. However, looking at the samples for this iteration of the model, while having some concepts in common with the hypothesis, the sentences in general are more nonsensical in comparison to those trained with the auxiliary loss (See an example in Figure 3.6).

The confusion matrix produced when evaluating our best model (MOSM, $N = 1$) on DIIN shows us where the classification model and our generative model agree (See Table 3.2). In our Random experiments, we find that the model has a bias towards predicting contradictions. This is observed here as well, with contradictions being the category with the highest agreement. We therefore cannot conclude that contradictions are easier for our model to generate. Also, using the original test set, the category in which DIIN performs the best is entailment, with a precision of 89.1% compared to 84.3% for neutral and 88.4% for contradiction. This suggests that generating suitable premises that entail the hypothesis is the hardest task for the model.

We also want to study how the classifier component of our model affects the generation of good samples. As shown in Figure 3.4, "Z precision" is higher then

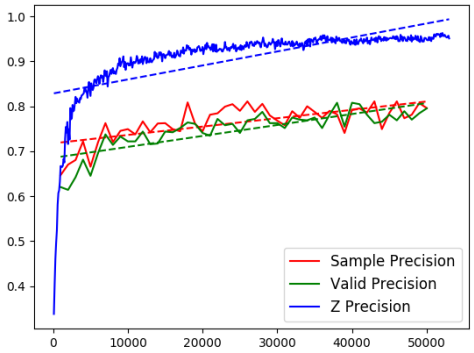


Figure 3.4 – Different classification precisions given by our classifier in our model (MOSM, N=10) during training. Sample Precision shows the probability that classifier predicts correct label for generated premise and related real hypothesis. Valid precision shows the probability that classifier predicts correct label for real premise and real hypothesis. Z precision shows the probability that the feedforward network $f_{\text{classifier}}(\mathbf{z}, z_{\boldsymbol{\eta}})$ predicts correct label ℓ, for given $\boldsymbol{\eta}$, ℓ and $\mathbf{z}$ drawn from prior $p(\mathbf{z}|\boldsymbol{\eta}, \ell)$.

0.9. This suggests that the classifier provides a strong regularization signal to the sentence representation $\mathbf{z}$. Because the autoencoder is not perfect, we do not observe the the same sample classification precision after $\mathbf{z}$ is decoded. However, we still observe a synchronous improvement of both sample and valid precision. It is therefore reasonable to expect that a better classifier and a better autoencoder would result in better generated premises.

3.5.4 Diversity Evaluation

In order to evaluate the diversity of samples given by our model, we compute the BLEU score between to premises generated conditioned on the same hypothesis and label. In other words, given a triple $(\boldsymbol{\phi}_i, \boldsymbol{\eta}_i, \ell_i)$ from test set, we draw two different samples $(\mathbf{z}_{i1}, \mathbf{z}_{i2})$ from the prior distribution $p(\mathbf{z}|\boldsymbol{\eta}_i, \ell_i)$. Then the decoder generates two premises $(\boldsymbol{\phi}_{i1}, \boldsymbol{\phi}_{i2})$ using greedy search conditioned on $(\mathbf{z}_{i1}, \mathbf{z}_{i2})$ respectively. The similarity score between generated premises is then estimated by:

$$\text{BLEU}_i = \frac{1}{2}\left(\text{BLEU}(\mathbf{z}_{i1}, \mathbf{z}_{i2}) + \text{BLEU}(\mathbf{z}_{i2}, \mathbf{z}_{i1})\right). \tag{3.36}$$

For comparison, we also compute the BLEU score between real premise and generated premise $(\boldsymbol{\phi}_i, \boldsymbol{\phi}_{i1})$. The average of diversity score between two generated premises is

Table 3.3 – BLEU score for different models

Model	BLEU_{RS}	BLEU_{SS}
Baseline (Mean)	14.4	N/A
Baseline (MOSM)	14.7	N/A
MOSM (N=1, -classifier)	14.4	46.7
MOSM (-auxiliary loss)	10.3	14.8
Mean (N=1)	11.9	27.9
Mean (N=10)	11.3	17.3
MOSM (N=1)	14.2	38.9
MOSM (N=10)	13.2	22.5

noted as BLEU_{SS}, the one between real and generated premises is noted as BLEU_{RS}. Since it is not to necessary have n-gram matches between premises, the BLEU score can be inaccurate on some data points. We employ the Smoothing technique 2 described in Chen and Cherry (2014).

As shown in Table 3.3, when we increase the number of samples N in the auxiliary loss, the diversity of samples increases for both mean pooling and MOSM. This can serve as empirical evidence that the diversity of our model can be controlled by choosing a different hyper-parameter N. The higher BLEU_{RS} given by MOSM method could be interpreted as the real premise being closer to the center of mass of prior distribution. We also observe a gap between BLEU_{RS} and BLEU_{SS}. The gap shows that the sampled premises are still relatively similar between themselves. After removing the classifier, we observe an increase in BLEU_{SS}. One possible explanation is that the classifier prevents the prior from overfitting the training data. We observe a decrease in both BLEU scores, after removing the the auxiliary loss. However, Table 3.1 and Figure 3.6 shows that removing the auxiliary loss give low quality samples.

While the auxiliary loss is essential for the prior and the decoder to learn to cooperate, using an auxiliary loss where $(N = 1)$ will collapse the prior distribution; instead of a distribution, the prior will instead learn to ignore the random input and deterministically predict $\mathbf{z}$. As shown in Figure 3.5, the auxiliary loss $(N = 10)$ only passes gradients to the $\mathbf{z}$s in the left region of the distribution. As a result, samples drawn from right region have a significantly lower chance to receive gradients from decoder, while the entire region receives gradients from the discriminator and

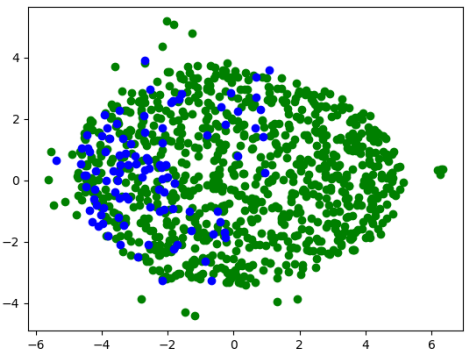


Figure 3.5 – Visualization of the effect of auxiliary loss with multiple samples. For a pair of $(\boldsymbol{\phi}, \boldsymbol{\eta})$, we repeat 100 times the process of compute auxiliary loss (N=10) in Equation 3.26. Blue points represent $\mathbf{z}_i$ selected by Equation 3.26, green points represent $\mathbf{z}_i$ that are not selected. Our model (MOSM, N=10) is used for computing $\mathbf{z}$ and perplexities. t-SNE is used to visualize high-dimensional data (Maaten and Hinton, 2008).

classifier. Therefore, the prior distribution can expand to more regions, but only those regulated by discriminator and classifier. This will increase the diversity of samples. However, we also observe that the precision slightly decreases in Table 3.1. This suggests that there is a trade-off in regularising the distribution and the precision on classification of the generated sentences.

3.5.5 Samples

Figure 3.6 shows several examples generated by our model (MOSM, N=10). These example shows that our model can generate a variety of different premise while keeping the correct semantic relation. Some of subjects in hypothesis are correctly replace by synonyms (e.g. "jockey" is replaced by "horse rider", "human" is replaced by "person" and "woman"). The model also get some potential logical relation correct (e.g. "reading a book" is contradicted by "with his hands in his pockets", "stands over a bread display" can either means "washing a window" or "preparing food in a kitchen").

However, we also observe that the model tries to add "a blue shirt" for most "man"s in the sentences, which is one of the easiest way to add extra information into the model. The phenomenon aligned with well-known *model collapse* failure case for most adversarial training based method. This observation gives an explanation for

Samples from MOSM (N=10)

H: a worker stands over a bread display .
L: Entailment
S1: a man in a blue shirt is preparing food in a kitchen .
S2: a man in a blue shirt is washing a window .

H: there is a jockey riding a horse .
L: Entailment
S1: a horse rider on a bucking horse .
S2: a jockey riding a horse in a rodeo .

H: a man sitting on the couch reading a book .
L: Contradiction
S1: a man is sitting on a bench with his hands in his pockets .
S2: a man in a blue shirt is standing in front of a store .

H: a baby in his stroller outside .
L: Contradiction
S1: a woman is sitting on a bench next to a baby .
S2: a woman is sitting on a bench in a park .

H: the man is being watched .
L: Neutral
S1: a man jumps from a bridge for an elderly couple at a beach .
S2: a man in a blue shirt is standing in front of a building .

H: there is a human selling hot dogs .
L: Neutral
S1: a person is standing in front of a food cart .
S2: a woman in a white shirt is standing in front of a counter selling food .

Samples from MOSM (-auxiliary loss)

H: a restaurant prepares for a busy day .
L: Neutral
S1: a pink teenager prepares on a tune on the roots .
S2: a UNK restaurant dryer for a canvas .

Figure 3.6 – Example sentence generated by our model (MOSM, N=10). **H** is the hypothesis, **L** is the label, **S1** is the first sample, and **S2** is the second sample. The samples shown below the line are drawn from a model trained without the auxiliary loss.

the relatively higher BLEU score between samples. The model also has some bias while generating premises (e.g. when the hypothesis mentions "a baby", the premise automatically mentions "a woman"), which aligns with the recent discovery that visual recognition tasks model tend to output biased predictions (Zhao et al., 2017).

3.6 Discussion

The broader vision of our project is to attain logical control for language, which we believe will allow us to perform better across many natural language

applications. This is most easily achieved at the word-level, by adding or removing specific words to a sentence, using word generation rules based on language-specific grammars. However, just as distributed word representations can be meaningfully combined Mikolov et al. (2013) with good outcomes, we believe that sentence-level representations are the way forward for manipulation of text.

The kind of control we seek to model, specifically, is characterized by the logical relationships between sentence pairs.Controlling semantic representation by modeling logical relationship between the input and output sentences has many potential use cases. Returning to the task of multi-document summarization discussed in the introduction, operating in the semantic space allows one to abstract the information of a document.Controlling the logical relationships among sentences provides a new way to think about what a summary is. Ideally, when multiple sources of information are given, we would like the output summary $\boldsymbol{\phi}$ generated by a machine to be entailable by the union of inputs $(\cup_{j\in\mathcal{J}}\boldsymbol{\eta}_j) \models \boldsymbol{\phi}$[3]. This addresses the problem of precision: the resulting summary now has a subset of the information available in the union of all the given hypotheses.

To address the problem of recall, we need the resulting summary to entail each one of the individual hypotheses: $\wedge_i(\boldsymbol{\phi} \models \boldsymbol{\eta}_j)$ Together, these two criteria form a clear formal definition for multi-document summarization,

$$\{\, \boldsymbol{\phi} \,:\, \wedge_i(\boldsymbol{\phi} \models \boldsymbol{\eta}_j) \,\wedge\, (\cup_{j\in\mathcal{J}}\boldsymbol{\eta}_j) \models \boldsymbol{\phi} \,\}$$

which represents the set of all possible $\boldsymbol{\phi}$ that fit the criteria.

In our paper, we toyed with the possibility of modeling the set $\{\, \boldsymbol{\phi} \,:\, \boldsymbol{\phi} \models \boldsymbol{\eta} \,\}$ by training a model with a distribution over different premises in the latent space $\mathbf{z}$. A good subsequent step would be modelling the first part of our logical description of multi-document summarisation,

$$\{\, \boldsymbol{\phi} \,:\, \wedge_{i\in\mathcal{J}}(\boldsymbol{\phi} \models \boldsymbol{\eta}_j) \,\} = \cap_{i\in\mathcal{J}}\{\, \boldsymbol{\phi} \,:\, \boldsymbol{\phi} \models \boldsymbol{\eta}_j \,\}$$

This suggests a possible avenue for producing such a premise is finding the intersection of the distribution over $\mathbf{z}$ for two given hypotheses that are likely enough to occur.

Future work can explore the possibility of this and determining the union of the

3. Here we assume there are no conflicting details.

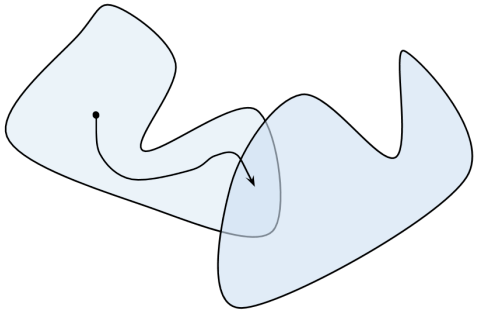


Figure 3.7 – Traversing the latent space to a common representation

hypotheses entailing the given premise.

3.7 Epilogue

The original goal for the project was to be able to synthesise a single document or sentence that would summarise all of the salient points in each individual sentence given. The idea was that once we had a continuous latent representation of the sentence, we could traverse the latent space to find a common region where the given $\mathbf{z}$ would entail two hypotheses simultaneously $(\mathbf{h}_1, \mathbf{h}_2)$. Figure 3.7 illustrates the idea graphically. More formally, we want to find $\mathbf{z}$ such that:

$$\begin{aligned}\arg\max_{\mathbf{z}} \{&D(\mathbf{z}, \mathbf{h}_1, \text{entail}) + D(\mathbf{z}, \mathbf{h}_2, \text{entail}) \\ &+ p(\ell = \text{entail}|\mathbf{z}, \mathbf{h}_1) + p(\ell = \text{entail}|\mathbf{z}, \mathbf{h}_2)\}\end{aligned}$$

Unfortunately, the sentences we tried to generate from this method did not have the properties we wanted.

The implicit assumption present here is that the distribution over the latent space represents possible elaborations over a sentence with less information. For example, under this assumption, "a dog sat on the ground" would produce a distribution that would contain a more informative sentence such as: "a brown dog sat on the muddy ground". It is not clear, however, from our method of training, that this would be a

property of the latent space that the model learns. Nevertheless, we find this idea an interesting one to explore as it gives us a way to meaningfully combine and think about the latent space of sentences.

4 Conclusion

Recent results demonstrate that unsupervised learning on language is still crucial to improving Natural Language Understanding tasks in machine learning. While significant improvements have been made from using heavy regularisation on language models (Merity et al., 2017), a more expressive output distribution (Yang et al., 2018), and improving performance for recent words, we believe that modelling global context has benefits for both language modelling and extracting meaningful representations for sentences. In this report we have given baselines on the Penn Treebank dataset language modelling task using PLVLMs.

We have preliminary results (Chapter 2) showing that with the right constraints on the decoder architecture, we are able to achieve better sentence-modelling results with the sentence-level latent variable. However, what is modelled by the latent variable leaves much to be desired. There is a tendency to capture prefix phrases of the sentence, and the generated sentences do not seem more coherent than the sentences generated by standard RNN LM. One avenue for future work would be to objectively evaluate the usefulness of these latent representations by using separate discriminative tasks.

Part of the hope of using latent variable language models is the separation of the syntax from semantics of the sentence. If the decoder was purely syntax aware, then the latent variable would have to model the semantics of the sentence. However, because natural language changes over time with use and cross-pollination from *other* languages, not only would vocabularies change, but syntax may also change, and we would do well to have capabilities to learn language structure from available data, as opposed to pre-determining a set of fixed rules agreed upon by linguists.

So then, how do we disentangle the two? For a start, translating a sentence from one language to another requires different grammatical knowledge of both the source language and the target language. This may provide a way to distill syntactic knowledge from the meaning of the sentence. The Vauquois Triangle (Figure 4.1) graphically illustrates this idea of going "up" the semantic hierarchy in translation,

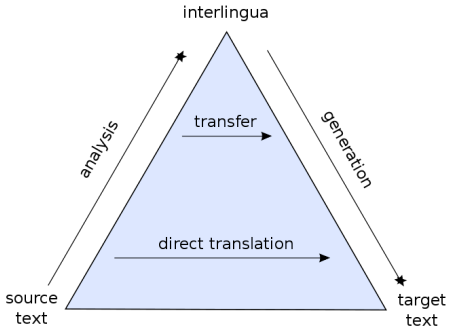


Figure 4.1 – Vauquois Triangle

with the ideal being semantic transfer at the apex of the triangle. Alternatively, there may be a way to deal with this within the same language, via paraphrases or logically entailing sentences, as discussed in Chapter 3.

Unfortunately, in translation, if both languages share a lot of structural similarity (e.g. English and French), a system that does this may be doing something much more superficial. Similarly, it is possible for a technique that relies on logically entailing sentences to fail to learn the right semantics, by simply learning superficial word similarities.

Through evaluating them with coreference resolution tasks, Trinh and Le (2018) show that using a large language model trained on a large dataset, has at least a limited understanding of language. For us, this raises several interesting questions: If this is achievable by large amounts of data, could the right model of language with the right inductive biases could perform the same task with less data? What other analyses could we perform on language models to figure out what other intrinsic features of language it has learned? To what extent can we learn *without* external labelled data to understand language?

We hope to explore in future work, not only the possible technical improvements in order to improve language modelling discussed in Section 2.3.3, and Section 3.6, but also the broader issues elaborated above.

Bibliography

Bahdanau, D., T. Bosc, S. Jastrzębski, E. Grefenstette, P. Vincent, and Y. Bengio (2017). Learning to compute word embeddings on the fly. *arXiv preprint arXiv:1706.00286*.

Bahdanau, D., K. Cho, and Y. Bengio (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bahl, L. R., F. Jelinek, and R. L. Mercer (1990). A maximum likelihood approach to continuous speech recognition. In *Readings in speech recognition*, pp. 308–319. Elsevier.

Bengio, Y., R. Ducharme, P. Vincent, and C. Jauvin (2003). A neural probabilistic language model. *Journal of machine learning research 3*(Feb), 1137–1155.

Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM 55*(4), 77–84.

Bouchacourt, D., R. Tomioka, and S. Nowozin (2017). Multi-level variational autoencoder: Learning disentangled representations from grouped observations. *arXiv preprint arXiv:1705.08841*.

Bowman, S. R., G. Angeli, C. Potts, and C. D. Manning (2015). A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Bowman, S. R., L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio (2015). Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Burda, Y., R. Grosse, and R. Salakhutdinov (2015). Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*.

Chelba, C., T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, and T. Robinson (2013). One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Chen, B. and C. Cherry (2014). A systematic comparison of smoothing techniques for sentence-level bleu. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pp. 362–367.

Chen, Q., X. Zhu, Z.-H. Ling, S. Wei, H. Jiang, and D. Inkpen (2017). Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Volume 1, pp. 1657–1668.

Chen, X., D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel (2016). Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*.

Chung, J., K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio (2015). A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pp. 2980–2988.

Conneau, A., D. Kiela, H. Schwenk, L. Barrault, and A. Bordes (2017). Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Davidson, T. R., L. Falorsi, N. De Cao, T. Kipf, and J. M. Tomczak (2018). Hyperspherical variational auto-encoders. *arXiv preprint arXiv:1804.00891*.

Dieng, A. B., Y. Kim, A. M. Rush, and D. M. Blei (2018). Avoiding latent variable collapse with generative skip models. *arXiv preprint arXiv:1807.04863*.

Dinh, L., J. Sohl-Dickstein, and S. Bengio (2016). Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*.

Edwards, H. and A. Storkey (2016). Towards a neural statistician. *arXiv preprint arXiv:1606.02185*.

Gal, Y. and Z. Ghahramani (2016). A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pp. 1019–1027.

Germain, M., K. Gregor, I. Murray, and H. Larochelle (2015). Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pp. 881–889.

Gong, Y., H. Luo, and J. Zhang (2017). Natural language inference over interaction space. *arXiv preprint arXiv:1709.04348*.

Grave, E., A. Joulin, and N. Usunier (2016). Improving neural language models with a continuous cache. *arXiv preprint arXiv:1612.04426*.

Gulrajani, I., K. Kumar, F. Ahmed, A. A. Taiga, F. Visin, D. Vazquez, and A. Courville (2016). Pixelvae: A latent variable model for natural images. *arXiv preprint arXiv:1611.05013*.

Gupta, A., A. Agarwal, P. Singh, and P. Rai (2017). A deep generative framework for paraphrase generation. *arXiv preprint arXiv:1709.05074*.

Guu, K., T. B. Hashimoto, Y. Oren, and P. Liang (2017). Generating sentences by editing prototypes. *arXiv preprint arXiv:1709.08878*.

Hsu, W.-N., Y. Zhang, and J. Glass (2017). Learning latent representations for speech generation and transformation. *arXiv preprint arXiv:1704.04222*.

Hu, Z., Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing (2017). Toward controlled generation of text. In *International Conference on Machine Learning*, pp. 1587–1596.

Huang, C.-W., D. Krueger, A. Lacoste, and A. Courville (2018). Neural autoregressive flows. *arXiv preprint arXiv:1804.00779*.

Huang, C.-W., A. Touati, L. Dinh, M. Drozdzal, M. Havaei, L. Charlin, and A. Courville (2017). Learnable explicit density for continuous latent space and variational inference. *arXiv preprint arXiv:1710.02248*.

Jelinek, F., B. Merialdo, S. Roukos, and M. Strauss (1991). A dynamic language model for speech recognition. In *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19-22, 1991*.

Jordan, M. I., Z. Ghahramani, T. S. Jaakkola, and L. K. Saul (1999). An introduction to variational methods for graphical models. *Machine learning 37*(2), 183–233.

Kingma, D. P. and J. Ba (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P., T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling (2016). Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pp. 4743–4751.

Kingma, D. P., T. Salimans, and M. Welling (2016). Improving variational inference with inverse autoregressive flow. *arXiv preprint arXiv:1606.04934*.

Kingma, D. P. and M. Welling (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kiros, R., Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler (2015). Skip-thought vectors. In *Advances in neural information processing systems*, pp. 3294–3302.

Kneser, R. and H. Ney (1995). Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, Volume 1, pp. 181–184. IEEE.

Koehn, P., F. J. Och, and D. Marcu (2003). Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pp. 48–54. Association for Computational Linguistics.

Kolesnyk, V., T. Rocktäschel, and S. Riedel (2016). Generating natural language inference chains. *arXiv preprint arXiv:1606.01404*.

Krueger, D., C.-W. Huang, R. Islam, R. Turner, A. Lacoste, and A. Courville (2017). Bayesian hypernetworks. *arXiv preprint arXiv:1710.04759*.

Maaløe, L., C. K. Sønderby, S. K. Sønderby, and O. Winther (2016). Auxiliary deep generative models. *arXiv preprint arXiv:1602.05473*.

Maaten, L. v. d. and G. Hinton (2008). Visualizing data using t-sne. *Journal of machine learning research 9*(Nov), 2579–2605.

Makhzani, A., J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey (2015). Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

Merity, S., N. S. Keskar, and R. Socher (2017). Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*.

Miao, Y. and P. Blunsom (2016). Language as a latent variable: Discrete generative models for sentence compression. *arXiv preprint arXiv:1609.07317*.

Mikolov, T. (2012). Statistical language models based on neural networks. *Presentation at Google, Mountain View, 2nd April*.

Mikolov, T., M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur (2010). Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, and J. Dean (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119.

Mou, L., Z. Meng, R. Yan, G. Li, Y. Xu, L. Zhang, and Z. Jin (2016). How transferable are neural networks in nlp applications? *arXiv preprint arXiv:1603.06111*.

Oord, A. v. d., N. Kalchbrenner, and K. Kavukcuoglu (2016). Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*.

Paszke, A., S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer (2017). Automatic differentiation in pytorch.

Peters, M. E., M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Polyak, B. T. and A. B. Juditsky (1992). Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization 30*(4), 838–855.

Press, O., A. Bar, B. Bogin, J. Berant, and L. Wolf (2017). Language generation with recurrent generative adversarial networks without pre-training. *arXiv preprint arXiv:1706.01399*.

Radford, A., K. Narasimhan, T. Salimans, and I. Sutskever (2018). Improving language understanding by generative pre-training.

Rajeswar, S., S. Subramanian, F. Dutil, C. Pal, and A. Courville (2017). Adversarial generation of natural language. *arXiv preprint arXiv:1705.10929*.

Rezende, D. J. and S. Mohamed (2015). Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*.

Rezende, D. J., S. Mohamed, and D. Wierstra (2014). Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*.

Salimans, T., I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen (2016). Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pp. 2234–2242.

Salimans, T. and D. P. Kingma (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 901–909.

Schuster, M. and K. K. Paliwal (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing 45*(11), 2673–2681.

Serban, I. V., T. Klinger, G. Tesauro, K. Talamadupula, B. Zhou, Y. Bengio, and A. C. Courville (2017). Multiresolution recurrent neural networks: An application to dialogue response generation. In *AAAI*, pp. 3288–3294.

Serban, I. V., I. Ororbia, G. Alexander, J. Pineau, and A. Courville (2016). Multimodal variational encoder-decoders. *arXiv preprint arXiv:1612.00377*.

Shabanian, S., D. Arpit, A. Trischler, and Y. Bengio (2017). Variational bi-lstms. *arXiv preprint arXiv:1711.05717*.

Shen, Y., S. Tan, C.-W. Huang, and A. Courville (2018). Generating contradictory, neutral, and entailing sentences. *arXiv preprint arXiv:1803.02710*.

Sohl-Dickstein, J. and D. P. Kingma (2015). Note on equivalence between recurrent neural network time series models and variational bayesian models. *arXiv preprint arXiv:1504.08025*.

Sønderby, C. K., T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther (2016). Ladder variational autoencoders. In *Advances in Neural Information Processing Systems*, pp. 3738–3746.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research 15*(1), 1929–1958.

Tan, S., C. J. Pal, and A. Courville (2018). Inferring identity factors for grouped examples.

Tiedemann, J. (2018). Emerging language spaces learned from massively multilingual corpora. *arXiv preprint arXiv:1802.00273*.

Tomczak, J. M. and M. Welling (2016). Improving variational auto-encoders using householder flow. *arXiv preprint arXiv:1611.09630*.

Tomczak, J. M. and M. Welling (2017). Vae with a vampprior. *arXiv preprint arXiv:1705.07120*.

Trinh, T. H. and Q. V. Le (2018). A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008.

Wainwright, M. J., M. I. Jordan, et al. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning 1*(1–2), 1–305.

Wan, L., M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus (2013). Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, pp. 1058–1066.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning 8*(3-4), 229–256.

Yang, Z., Z. Dai, R. Salakhutdinov, and W. W. Cohen (2018). Breaking the softmax bottleneck: A high-rank rnn language model.

Yang, Z., Z. Hu, R. Salakhutdinov, and T. Berg-Kirkpatrick (2017). Improved variational autoencoders for text modeling using dilated convolutions. In *International Conference on Machine Learning*, pp. 3881–3890.

Yu, L., W. Zhang, J. Wang, and Y. Yu (2017). Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pp. 2852–2858.

Zhang, Y., Z. Gan, and L. Carin (2016). Generating text via adversarial training. In *NIPS workshop on Adversarial Training*, Volume 21.

Zhang, Z., Y. Song, and H. Qi (2017). Age progression/regression by conditional adversarial autoencoder. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Volume 2.

Zhao, J., T. Wang, M. Yatskar, V. Ordonez, and K.-W. Chang (2017). Men also like shopping: Reducing gender bias amplification using corpus-level constraints. *arXiv preprint arXiv:1707.09457*.